STUDIES OF DPA FLUORESCENCE ENHANCEMENT

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Medical Physics

Raphael Nolden BSc

Department of Physics and Astronomy

University of Canterbury

Christchurch, New Zealand

2007

Supervisor: Associate Professor Lou Reinisch PhD

Table of Contents

List of Tables

Table of Figures

1 Acknowledgements

First and foremost, I wish to thank my supervisor Assoc. Prof. Lou Reinisch for his amazing insight, patience and gentle yet persistent guidance throughout my research. He started this line of work in the department and encouraged me to take part in it. I also wish to thank my co-supervisors, Dr. Richard Watts and Assoc. Prof. Peter Cottrell for their assistance. Also the many technical staff of the department who have been very helpful and always ready to give me the support I needed. I am very grateful for the financial assistance provided by Veritide Ltd. during this research.

I wish thank my family, especially my parents, for their continued encouragement throughout the last year. Special mention must also go to my brother, Sandy, who painstakingly proofread this thesis and corrected all of my grammatical errors. I also wish to thank my flatmates and many friends for all their support, encouragement, and understanding in times of extreme stress and exhaustion.

2 Abstract

The processes involved in the enhancement of the fluorescence profile of dipicolinic acid (DPA) were measured and analysed, with particular emphasis on their potential application to the rapid identification of suspicious powders. The research was conducted in contribution to the anthrax detector currently under development at this department. Using the enhancement of fluorescence as a method of determining whether a sample contains spores shows great potential because DPA is not found in most powders that do not contain spores. Thus, its detection is a good indication of the presence of spores.

The research presented in this thesis primarily focuses on the optimisation of measurement and enhancement techniques. Both DPA and milk powder (containing spores) were used as anthrax simulants. We found that 210 nm light was the optimal wavelength for the enhancement of DPA; however, as most light sources have a higher intensity at longer wavelengths, the use of 270 nm light may be more effective. At low concentrations, there is a linear relationship between detected fluorescence intensity and the quantity of DPA present. A linear response was also found to the enhancement-light exposure time.

Index heading: Bacteria, Spores, Bacteria identification, detection, fluorescence, fluorescence enhancement,

3 Glossary

2,6-Pyridinedicarboxylic acid – commonly referred to as dipicolinic acid. A chemical compound found in Bacillus anthracis spores and used in this research to simulate these. Abbreviated as DPA in this thesis.

Anthrax – the disease caused by the bacterium Bacillus anthracis. Also commonly used to describe the bacterium and/or its spores.

Bacillus anthracis – a rod-shaped gram positive bacterium of the Bacillus group. Bacillus is one of many groups of bacteria, including Bacillus, Clostridium, Sporohalobacter, Anaerobacter, and Heliobacterium, which form spores. Its spores are found in soil throughout the world, and can be used as a bio-terrorism/warfare agent.

CaDPA – Calcium DPA: chemical compound created when a DPA solution is made in tap water containing calcium.

Dipicolinic acid - 2,6-Pyridinedicarboxylic acid.

DPA – 2,6-Pyridinedicarboxylic acid.

Emission profile – describes how effectively a substance is enhanced by a given wavelength. It is calculated by enhancing a substance with various

wavelengths of light, and comparing the level of enhancement to the wavelength.

Enhancement – the change in the fluorescence of a sample which occurs when it is illuminated with light.

Fluorescence – an optical process where a molecule absorbs a photon of one wavelength or energy and then emits another photon of longer wavelength or lower energy.

Spore – a very stable and hardy stage in the life cycle of certain bacteria, such as anthrax. Spores can withstand UV and Gamma rays, alcohol, heat, chemical disinfectants, and Desiccation. They can remain in this state for centuries because their metabolism stops. Once conditions become favourable they will hatch and become active again. Sometimes referred to as an endospore.

Abbreviations

Monochromators – monochromatic illuminators.

OD – optical density

PMT - Photomultiplier tube

4 Introduction

Bacteria are unicellular micro-organisms. They are the cause of many fermentations, transformations, (of both organic and inorganic materials), and most importantly, in the context of this thesis, infectious diseases [1]. The nature and severity of these diseases range from harmless afflictions, to severe and often fatal, (particularly in an historical context), diseases such as tetanus, leprosy, and anthrax. The creation of spores by some species, such as *Bacillus anthracis*, exacerbates efforts to eradicate them, because they are almost indestructible in this state.

Anthrax is caused by *Bacillus anthracis*, a gram-positive bacterium commonly found in soil throughout the world [2]. The pathogenesis of anthrax proceeds as follows: After the spores enter the body of the victim either by ingestion, inhalation or through an open wound, they are phagocytosed by macrophages and carried to lymph nodes [2]. Once there, the spores germinate inside the macrophages and become vegetative bacteria [2]. These are then released, and multiply in the lymphatic system before entering the blood stream, causing concentrations as high as 10^8 organisms/mL of blood [2]. No further immune response occurs after the initial one [2]. Unless treated, the host is usually killed by the septicaemia that results from the high concentrations of bacteria, and toxaemia caused by the toxins they release into the bloodstream [2]. After

the host's death, conditions for spore-forming bacteria begin to deteriorate, so the cell responds by starting the sporulation process [3]. The first stage of this is a unique form of cell division, giving rise to one larger cell called the mother, and a smaller one called the forespore [3]. The mother cell engulfs the forespore, yet both maintain complete genomes [3], despite one being inside the other. Bacterial spores are often referred to as endospores because they form inside the mother cell [3]. The spore that finally emerges is comprised of a protoplast, protoplast membrane, cortex and three coat layers [4]. In this spore state it can remain dormant until conditions become favourable again, usually once it has passed into the body of another host.

Spores can survive in this metabolically dormant state for decades [2], surviving in a laboratory at room temperature for 60 years [5], and have been recovered from bones thought to be 200 years old [5]. They are also resistant to a diverse range of environmental hazards, including heat [2-6], cold [5], humidity [5], long dry periods [2, 5], freezing [3], freeze-drying [3], high pressure (≥12000 atm) [3], desiccation [3, 4], chemical disinfectants [2, 5], ultraviolet light [2, 3], and gamma radiation [2, 3].

Because of the spores' metabolic dormancy and their resistance to harsh environmental factors, they are able to survive in the ground for very long periods of time. This makes them very difficult to eradicate in nature.

Historically, most cases of anthrax in humans have been contracted from interactions with infected livestock or animal products [6]. Human to human infection is unknown [2]. Comprehensive immunisation programmes have reduced the incidence of anthrax in animal populations [7], though occasional localised outbreaks still occur. [5].

More recently, and thus more importantly in the context of this thesis, anthrax spores have been grown in laboratories for use in biological weapons. During WWII the British tested anthrax bombs on the Island of Gruinard off the north-west coast of Scotland [5, 8], by exploding many small devices containing anthrax [5]. The island was decontaminated over 40 years later [5]. Many other countries have developed weapons using anthrax [2], including Russia [9, 10], Canada [10], the United States [10], and Japan [2, 10], the only nation to have used an anthrax weapon in combat [2]. This occurred during 1940, in Manchuria. [2].

Some of the properties of *Bacillus anthracis* spores make them ideal as bio-terror agents. These properties include their extremely hardy nature, which means that once produced/obtained, no special equipment is needed to maintain the spores. They will also remain dormant, but alive, for long periods without the need for any specialised equipment, making them resistant to changes in environment that may be encountered on their way to the target.

Another very important characteristic is the lethality of anthrax; inhalation of a few thousand spores can be fatal [11]. The LD50 of anthrax is generally believed to be ~8000 spores [12].

One popular form of terror attack, using anthrax spores, is sending them to specific targets by post; the spores can be placed in an envelope, which is then sealed and posted. On opening of the envelope they will rapidly disperse widely into the surroundings, including the lungs of those unfortunate enough to be in the vicinity. The simplicity, ease of execution, and success of this method was demonstrated by the series of attacks that occurred in the United States in 2001.

In late 2001, anthrax-bearing letters were sent to several addresses in the US. They were all sent from Trenton, New Jersey [13]. Two were posted on the 18th of September and sent to news agencies (New York Post and NBC) [13], and the other two, posted on the 18th of October, were sent to senators, (Tom Daschle and Patrick Leahy) [13]. All four letters contained the same “ames” strain of anthrax [13], and were thought to be from one sender [13]. The letter sent to NBC was handled by various staff members before being opened [13]. The New York Times letter was not opened [13]. The letter to Senator Daschle was opened by an aide and all 28 people in the vicinity tested positive to anthrax, as did 7 of 25 in his office one floor below [13]. This letter was

believed to contain about 2 grams of powder, containing approximately 200 billion to 2 trillion spores [13]. The other letter was mis-sent due to an error made by an optical address-reader, and was found and analysed later. The spores were between 1 μm and 3 μm in size, and coated with fine particles of frothy silica glass [13].

The total number of people infected with anthrax from October to November 2001 by these attacks was 22 [5, 14]. Eleven suffered from inhalational anthrax [5, 14, 15] and the others from the less dangerous cutaneous form [5, 14]. Five of those with inhalational anthrax died [5, 14]. The majority (91%) of those infected were workers at the target businesses and mail handlers [14]. The remainder are thought to have been infected as result of cross-contamination between envelopes along the mail's processing and delivery route [14].

Decontamination costs are very high; the cost of cleaning up the Hart Senate Office Building on Capitol Hill, where one of the letters was opened, is estimated to have been US$27 Million [5]. The total cost of decontaminating all the affected sites is thought to have been hundreds of millions [5]. Many different substances and procedures were used in the clean up process, including paraformaldehyde, ethylene oxide, methyl bromide, vaporised hydrogen peroxide, peroxyacetic acid, chlorine dioxide gas, and liquid bleach

products containing sodium hypochlorite [5]. Usually the building was fumigated with chlorine dioxide for 8-48 hours [5], followed by localised applications of the other products as required [5].

The success of the genuine anthrax attacks, the huge amount of disturbance they caused, and the high levels of fear they instilled in the population, meant that everyone was on high alert of white powders. This led to many hoaxes; the majority of samples of white powders examined in relation to anthrax scares were found to be benign [16], usually consisting of some other harmless/common white powder such as artificial sweeteners (e.g. aspartame and saccharin), sugar (sucrose), wheat flour, corn starch, cleaning products, pain killers, etc. [17]. Many of the scares came in the form of a letter from an unknown source containing white powder. Other things were also used, as occurred at the offices of B'nai B'rith in Washington DC, where a Petri dish containing a red gelatinous substance was used [11]. Many hoax cases occurred before the attacks of 2001 [11], and immediately following them, many thousands more occurred [17].

On the 24th of April 1997, a small package was delivered the headquarters of B'nai B'rith in Washington DC. [11, 18]. The package was labelled "anthrachs" and "Anthracis Yersinia," and contained a red gelatinous substance [11]. The building and a nearby hotel were immediately isolated

[18] and over 100 people were quarantined onsite [11]. Thirty people, including emergency workers and members of the public, were decontaminated [11, 18]. The dish was secured and sent to a federal laboratory in Bethesda, Maryland. After nine hours, and the involvement of a plethora of emergency response teams, the laboratory declared the package to be non-hazardous [18]. It contained *Bacillus cereus*, a non toxic spore similar to anthrax [19].

Current methods employed in the identification of unknown bacteria, and thus white powders found in suspicious circumstances, are very time-consuming [20], involving complex series of tests before the bacteria, (if any), can be identified [4, 21].

Because these identification techniques are so slow, every discovery of suspicious white powder causes a costly, large-scale disturbance. The surrounding area needs to be isolated, and people decontaminated and treated, in case the attack is genuine. This all needs to happen well before the powder has been identified. The disruptions caused, and the number of emergency response organisations involved, make each incident very costly to deal with. When, for example, a white powder spills from an envelope at a large mail sorting centre, the entire centre is shut down until the sample has been tested. In most cases, the powder is found to be benign.

There is a need for a device that can quickly distinguish between real spores and other powders [16]. As many of the powders used in these hoaxes are common household substances, which do not contain any spores, being able to eliminate these is the highest priority. So this device would need the ability to dismiss samples that do not contain spores within minutes, so that no further investigation would need to take place. If the powder is found to contain bacterial spores, the usual methods of identification would be required. This elimination of non-bacterial powders from an investigation is currently very time-consuming because it is impossible to distinguish spores from other powders just by looking at them, even with the aid of a microscope. So a different approach needs to be employed. Rapid and conclusive identification of a characteristic that is unique to bacterial spores is required. Fortunately, bacterial spores possess such a characteristic – they contain 2,6-pyridinedicarboxylic acid [3].

2,6-Pyridinedicarboxylic acid, or as it is more commonly known, dipicolinic acid (DPA), in fact makes up approximately 5-15% of the dry weight of spores [3, 22, 23]. It is thought to contribute toward their heat resistance. [22, 23].

Apart from its uniqueness to spores, DPA has other interesting and, in our case, very useful properties. It fluoresces when excited by UV light; however, this in itself does not enable it to be distinguished, as many substances

fluoresce. The fluorescence of DPA, however, can be enhanced. This enhancement causes a dramatic increase in the fluorescence emitted. The enhancement is relatively easy to achieve, by exposing it to UV light of a shorter wavelength (200-300 nm).

To quickly determine if a sample contains spores or not, a detector could measure the fluorescence of a sample, enhance it, and re-measure the fluorescence to determine if it has changed. From this, it could calculate if DPA, and thus spores, are present.

To create such a detector, a better understanding of the processes that take place in the DPA specifically, and in the spores in general, is required. To gain this understanding, further research is vital. This includes methods of optimising all processes involved, such as finding the best wavelength for enhancing the DPA, and measuring the resulting fluorescence most effectively. This is important, because optimising these processes allows us to minimise the time required for each. Furthermore, because of the high-risk application it is intended for, a great deal of confidence is required in the methods employed; both accurate and consistent results must be obtained. Knowledge of other parameters and limitations is also required, including the minimum number of detectable spores, and the linearity of the response to

enhancement and variation in concentration. Also, processes that could distort the results, such as saturation, need to be quantified.

The research in presented in this thesis is a contribution towards this effort, and, more specifically, towards an anthrax detector being developed by members of the Department of Physics at the University of Canterbury.

Most of this introduction has focused on anthrax for two reasons: firstly, because of the recent attacks and the resulting realisation of the importance of new detector technology, and secondly, because this research was primarily done to contribute towards the anthrax detector being developed here. It is important to note, however, that this technology is in no way restricted to use in this area; it also has many applications in the food technology sector. Bacterial spores are a cause of food spoilage [4] and present a variety of problems to the food industry [3]. So, rapid testing for presence of bacterial spores, such as in milk, would be very useful. This could allow milk to be tested before its dehydration process was started, and simplify some areas of quality testing of fresh milk products. These areas currently have a need for improved technology, and what we are doing could contribute towards this. Finally, identification of unknown bacteria is important for many sectors, yet less that 1% of all bacteria can be identified using current technology [24, 25],

Using fluorescence may, with further research and development, eventually enable bacterial species to be identified.

The research described in this thesis was undertaken to better understand the enhancement process that occurs in DPA, and specifically how this could be utilised in the development of an anthrax detector. Many questions require answers before such a device could be made, and this research aimed to provide some of them. We focused on the following areas because of their fundamental importance to the understanding of the processes involved, as direct requisites for the development of the detector. We investigated the emission profile of DPA, which was important because it enabled the enhancement process to be optimised and better understood. We also verified the measurement protocols that could be employed in the detection process to ensure that these did not interfere with the sample's enhancement. Furthermore, we investigated the use of polarisers and how they could be employed to reduce scattered light, thereby improving the quality of the signal.

5 Materials and Methods

This chapter describes the various apparatus and experimental techniques employed throughout the course of the research that forms the integral part of this thesis. All results and their analyses will be dealt with separately in the following chapter.

5.1 Machines Used to Take Measurements

In this section we describe and give specifications of the apparatus used in the collection of data for the experiments performed as part of this research.

5.1.1 SLM 8000C Photon Counting Spectrofluorometer

The SLM 8000c Spectrofluorometer uses a 450 W xenon arc lamp as a light source. The instrument uses two aberration-corrected concave gratings as excitation monochromators. The gratings have 1,500 grooves/mm and reciprocal dispersion of 2 nm/mm. the input focal length is 0.32 m. The emission monochromator is the same as the input one, with the addition of a holographic grating. All slits were set to 4 mm bandwidth. PMTs, set to current mode, were used as the detectors.

5.1.2 GBC UV-Visible Cintra 40 Absorption Spectrometer

This machine was used to measure the absorption of samples before further experiments were carried out. This enables the concentration of the anthrax simulant to be determined. It has dual light sources, a deuterium lamp for the UV part of the spectrum (<300 nm), and a quartz-iodine lamp for the visible parts of the spectrum (>300 nm). The machine was set up to scan from 200 nm to 700 nm, measuring with a step size of 0.427 nm and a speed of 1,000 nm/min. The slit width was 2.0 mm and the operation mode used was absorption. It uses double monochromators to enable measurements of up to 4 OD to be taken.

The machine compares the samples in two cuvettes and measures the difference between them to calculate how much light is absorbed. Both cuvettes must be the same and for all experiments. Quartz cuvettes were used, to allow UV light to transmit without excessive absorption. The water-containing cuvette was placed in the rear slot, while the cuvette containing the sample to be measured was placed in the front slot, to ensure the readout was positive.

Figure 5.1: The Cintra 40 absorption spectrometer.

5.1.3 ISS PCI Photon-Counting Spectrofluorometer

Some measurements of the fluorescence emission were also done using the ISS PCI Photon-Counting Spectrofluorometer. This has a 300 W high-pressure arc lamp. The monochromators have a wavelength range of 200-900 nm, with an accuracy of ±0.2 nm and a slew rate of 160 nm/s. All lenses are UV-grade fused silica and polarisers are UV-grade Glan-Thompson with a length aperture ratio of 2.0. The optical design and geometry can be seen in Figure 5.2. It uses a parallel beam design and a short path length to optimise its results. The detectors are both PMTs, (Hamamatsu, Japan), with a wavelength range of 240-900 nm. These are linear up to 4 million counts/second, and have a signal-to-noise ratio of 2000:1. The device is connected to a computer running Vinci (Version 1.6.SP4), to control the machine and collect the data.

The following parameters were used when performing experiments: monochromator slit widths were varied to optimise the signal for the different

parts of this research but kept constant during each set of experiments. They varied from 0.5 mm to 2.0 mm. Experimental parameters of the software were also varied with maximum iterations of either 15 or 20. Most experiments used the raw channel measurement type but some were also done using intensity. The raw channel mode allowed full data analyses to be made, to compensate for the variations in enhancement-lamp intensity. (See appendices).

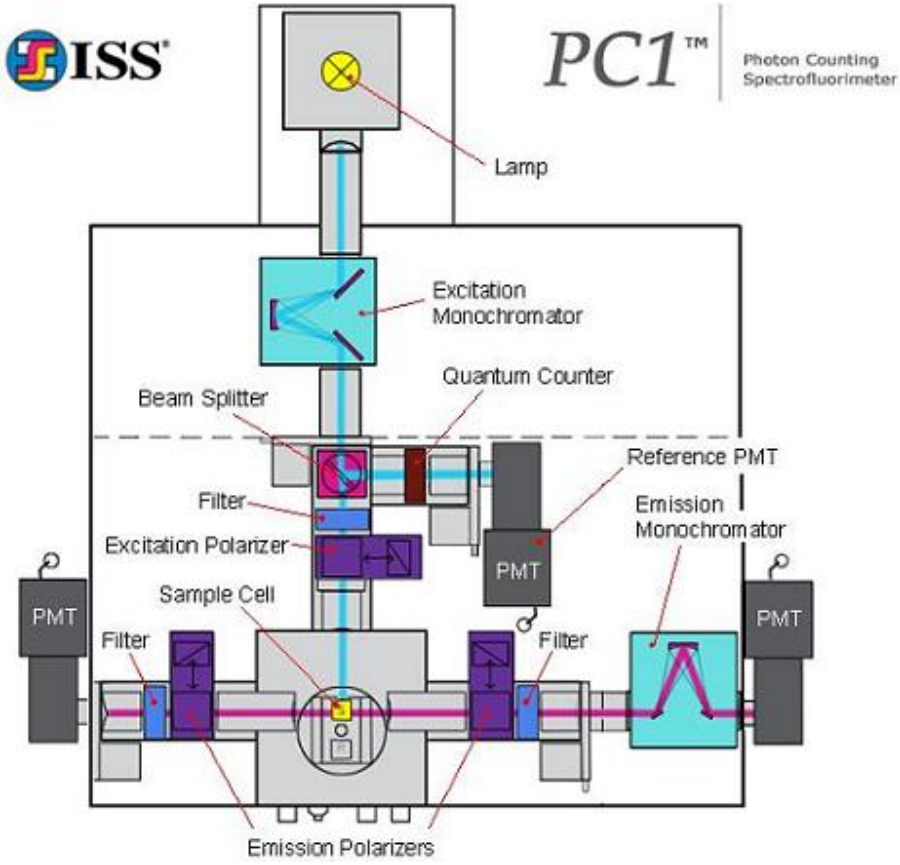

Figure 5.2: Schematic diagram of the optics of the ISS PCI Photon-Counting Spectrofluorometer [26]

Figure 5.3: ISS PCI Photon-Counting Spectrofluorometer

5.2 Lamps Used to Enhance the Sample

Many different lamps were used to enhance the samples; experiments were performed to determine the optimal lamp to use. However, the lamp used also depended on availability as some were only acquired later. Table 3 summarises the different lamps used.

Lamp Name/Description.	**Type**	**Spectrum**	**Output (W)**
Lamp in SLM 8000C Spectro fluorometer	Xenon arc lamp	Continuous	450
Water-cooled PTI High Energy Arc Lamp	Arc lamp	Continuous	150
Lamp in ISS PCI Photon Counting Spectro fluorometer	Xenon arc lamp	Continuous	300

Table 1: Comparison of lamps used for enhancement.

5.3 Anthrax Simulants

Anthrax is too dangerous to work with, (and unavailable), so other substances are used to simulate its required properties. The substances need to simulate the properties we are trying to detect fairly accurately. The various substances that were used as anthrax simulants are briefly outlined in this section with reasons given for why they were chosen.

5.3.1 Milk Powder

Milk powder can be used as an anthrax simulant because it contains the spores of other bacteria, such as *Bacillus stearothermophilus*, that have some important characteristics which are similar to those of anthrax. These are not harmful to humans in the quantities found in milk powder. Most importantly these bacteria also contain DPA which is what we use for our detection.

Milk powder contains DPA within a spore, mixed with other substances, as may be encountered in a real anthrax threat, making it a useful simulant. DPA is the chemical compound in isolation, without interferants and not contained within a spore, allowing certain of its properties to be studied in isolation without the presence of other substances that may interfere with the process of gathering relevant data.

5.3.2 2,6-Pyridinedicarboxylic Acid (DPA)

2,6-Pyridinedicarboxylic acid, 99%, is referred to in this thesis as DPA. It has the chemical symbol $C_7H_5NO_4$. We used DPA manufactured by Sigma-Aldrich, Batch 11411ED. As mentioned above, pure DPA was used in these experiments to enable us to focus on individual properties.

5.3.3 Summary of our Simulants and Comparison of their Properties

The details of the simulants used in these experiments are summarised on table Table 2.

Product Name	**Type**	**Manufacturer/Brand**
Whole milk powder	Whole milk powder	Unknown, from supermarket bulk bin.
Pam's Whole Milk Powder (400 g Sachet)	Whole milk powder	Pam's
2,6-Pyridinedicarboxylic acid (DPA)	DPA	Sigma-Aldrich

Table 2 Anthrax simulants used during the course of this research.

As mentioned in the previous section, both simulants have individual advantages, potential drawbacks and thus different uses. The characteristics of interest are summarised on Table 3.

Property	Milk Powder	DPA
Scattering from non fluorescing material	Yes	No
Simulates real spores by having other substances in it	Yes	No
Pure fluorescent substance	No	Yes
Other interferants	Yes	No
Absorption	Yes	Yes

Table 3: Comparison of the Anthrax simulants' properties.

5.4 Experimental Methodology

Before we started this research it was thought that the best wavelength to use for the enhancement of the DPA was 277 nm, based on the absorption peak of DPA at 277 nm. This number had to be checked, validated, and possibly quantified. Once was done, or a better wavelength discovered, other experiments could optimise the enhancement using this knowledge.

We also performed some experiments to ensure that the methods and apparatus used were functioning as required, and not creating errors in our data. We checked that our method of collecting the data was not causing any further enhancement of the sample, and ascertained the appropriate warm-up times of the apparatus.

Methods of optimising and refining the data-gathering process were explored, including making use of polarisers. This optimisation was achieved by the polarisers preferentially removing the light being scattered by the sample.

A further area of interest was the linearity of response to enhancement light. This was very important for validating the experiments, because we had to ensure that no saturation was occurring at any point during the process, e.g. because of too high concentrations or over-enhancement. Finding the linear range also allowed us to determine the useful range of machines employing this technology.

In the following section we describe the procedures and processes used to perform the experiments that form the integral part of this thesis. The results and discussion are dealt with separately in the following chapter.

5.4.1 Checking the Null Effect of 350 nm Light

The enhancement was calculated from the fluorescence emission profiles. These were measured whilst exciting the sample with a 350 nm light. We took this measurement twice, once before and once after enhancing the sample and subtracted the pre-enhancement value from the other to find the enhancement. Because we were exposing the sample to equal amounts of light during this measuring process and the enhancement stage, it was important to ensure that this wavelength of light was not having any enhancing effects on the sample. We checked this by means of the following experiment.

We created a solution of DPA in tap water by dissolving as much DPA (in crystal form) in 100 mL of water as possible. We transferred this solution into quartz cuvettes ensuring that no un-dissolved crystals were included and used these samples to take our measurements. Each measurement was taken from a fresh sample that was created immediately before use, placed in the Spectrofluorometer (ISS PCI), and enhanced for different lengths of time with 350 nm light. Whilst the original solution was not in use, its exposure to potentially enhancing light was minimised. After enhancement, the emission profile was measured. The initial measurement was taken with no enhancement, and, after all other measurements had been completed, a further non-enhanced measurement was performed to ensure the solution had not changed during the course of the experiment.

The measurements were taken using the intensity measurement-type. The enhancement and readout were performed with 350 nm light and the scan was done over a range of 360-700 nm in 1 nm increments.

5.4.2 Finding Appropriate Warm-Up Times for the Apparatus

Some of our early measurements of the emission profile of CaDPA were surprising; our data appeared to be dependent on the order in which it was measured.

This made us realise that something was causing some form of time-dependent effect. We were using warm-up times of at least 30 minutes but decided that this may not be enough. So we took a measurement of the machine's response by running it from cold and measuring how the intensity changed as the machine warmed up.

This initial experiment was set up using a water-filled quartz cuvette in the sample chamber. Its main role was to provide a consistent scatter medium. We set both the excitation and emission wavelengths to 350 nm so that we were just measuring the amount of light being scattered. We used 2 mm slits on the first monochromator, before the beam splitter, and 0.5 mm slits on the one following the sample chamber. To prevent the reference-PMT from saturating, we inserted a single piece of metal mesh beyond the Rhodamine dye to reduce the signal.

The experiment type used for this was slow kinetics, allowing measurements of intensity changes with time to be taken over long periods. The raw channels measurement type was used because this outputs the data from both PMTs and allows each to be analysed individually. The parameters used for this experiment are summarised on Table 4.

Parameter	Setting
First slit width	2 mm
Second slit width	0.5 mm
Mesh position	In optical path after dye
Cuvette	Quartz, water filled
Experiment type	Slow Kinetics
Measurement type	Raw channels
Maximum iterations	20
Time	32,400 seconds (9 hours)
Time interval	5 seconds

Table 4: Experimental parameters used for the spectrometer in this experiment.

From this first measurement we found that the machine warms up and the response flattens out after approximately 15,000 seconds (~4 hrs).

After finding that the effect was real, we decided to find out which components of the apparatus were causing this; and if it was caused by multiple components, we wanted to know how much each one was contributing. We ran a series of experiments with different combinations of components pre-warmed, to determine how much each was contributing. The combinations we ran are summarised on Table 5.

Experiment #	Lamp	Fluorometer	Computer
1	On	On	Off
2	On	On	On
3	On	On	On with shutter open
4	On	Off	Off
5	Off	On	Off

Table 5: Combinations of warmed-up and cold apparatus used to determine each component's contribution to the warm-up time.

We warmed up the components that were to be warm for long periods of time, to ensure they were ready, then turned on the other parts and began measuring as soon after this as possible. For these experiments, we used an empty quartz cuvette rather than a water-filled one. This was done to ensure the no bubbles were forming, substances settling out of solution or other such things changing and interfering with our measurements. Furthermore, the cuvettes were cleaned and dried using compressed air and had lids placed on them. We also suspected that the dye may have been bleaching due to over-exposure to light, so we put the mesh in front of the dye to reduce this exposure. We set up the machine using the parameters summarised in Table 6.

Parameter	Setting
First slit width	2 mm
Second slit width	1 mm
Mesh position	In optical path before dye
Cuvette	Quartz, air cleaned, filled and sealed
Experiment type	Slow Kinetics
Measurement type	Raw channels
Maximum iterations	15
Time	15,000 seconds
Time interval	5 seconds

Table 6: Experimental parameters used to determine the weight of each component's contribution to the warm-up time.

5.4.3 Emission Profile of DPA, and its Variation with Concentration

Our early measurements of the emission profile appeared to be inconsistent. When these inconsistencies persisted, and were greater than expected, even after eliminating the warm-up effects of the apparatus, discussed in the previous section, we investigated the possibility that this was caused by variations in the concentration. When preparing samples for different experiments, it is difficult to make them of exactly equal concentrations. This is because it is difficult to dissolve the crystalline DPA in water, so more is put in than required and the sample is mixed and stirred. It is allowed to settle before some of it is carefully removed, ensuring no un-dissolved crystals are in the final solution used for the experiments.

This series of experiments was undertaken to determine the optimal enhancement wavelength from the emission profile of CaDPA, and investigate if this varies with changes in concentration. Several samples were prepared with different concentrations of CaDPA in water, and for each concentration an enhancement was done, using a range of wavelengths (200-300 nm, in 10 nm increments). An emission profile could be determined for each wavelength and thus, by combining the data, an averaged emission profile was determined. From this, the optimal wavelength of light to be used when enhancing CaDPA could be ascertained.

The CaDPA solutions were created by dissolving DPA in tap water. Calcium DPA (CaDPA) forms because of the calcium present in the water. After dissolving as much DPA as possible in the water, some of the solution was carefully separated into another beaker, ensuring that none of the un-dissolved crystals were transferred. Small amounts of this sample were then removed using a pipette and added to a larger amount of water to create the final solution used to make each individual sample. From this solution 1 mL samples were extracted, placed into cuvettes and measurements taken immediately to ensure consistency. We put 1 mL of this solution into 1 cm quartz cuvettes, creating 1 cm^3 samples. The cuvette was placed on a specially prepared block to maximise the percentage of the sample exposed to the light,

thus minimising the possibility of the enhanced solution moving out of the light and not contributing to the signal. Throughout the procedure, samples were handled in such a way so as to minimise the amount of potentially enhancing light reaching them before they were placed in the apparatus.

Because it was very difficult to dissolve the DPA in water, the amount was not measured; instead, the concentration was determined as follows.

The Cintra Spectrometer was used to determine the absorption of the sample; the concentration of the CaDPA can be determined thus:

$$\text{Absorption} = -\text{Log}_{10}\left(\frac{I}{I_0}\right) = \alpha\, c\, l$$

Where:

I = the transmitted intensity

I_0 = the initial intensity (in this case the intensity of the other channel

α = the absorption coefficient (a constant for a given substance) for CaDPA this is 138,000 M^{-1} cm^{-1}

c = the concentration in Mol L^{-1}

l = the path length through the solution in cm

The machine compares two samples to determine the relative absorption between them. Both samples were contained in quartz cuvettes and the reference sample was tap water.

Since the initial aim of this part of the experiments was to determine the enhancement achieved by different wavelengths, samples were created and measured for all wavelengths from 200 nm to 300 nm, in 10 nm increments. The following procedure was used: The sample was mixed and placed into the Spectrofluorometer. A pre-enhancement measurement was taken using 350 nm light which causes no enhancement, (see previous section). This gives the background fluorescence. Then the machine was run twice at the desired enhancement-wavelength, exposing the sample to it. We ran this as a separate experiment to enable the machine to automatically terminate the enhancement after a predetermined time. After a post-enhancement measurement was taken, subtracting the pre-enhancement values from the post-enhancement values allowed the actual enhancement to be calculated.

These measurements were repeated for different concentrations of CaDPA to determine whether the enhancement changed in a linear way, and if the enhancement was directly proportional to the concentration.

Large amounts of data were collected during these experiments, and this, together with the level of data manipulation and analysis, required us to write a special computer programme. The first version analysed data for one given concentration; the code can be found in Appendix 2. Once multiple concentrations were being dealt with simultaneously and compared, a further,

more complicated, programme was needed, the code for which can be found in Appendix 3. Using these programmes, the data was analysed and the results determined. The single concentration programme is able to smooth data if required, corrects for variations in the intensity of the enhancing lamp, if desired, and performs other required data manipulations when required. It also finds the peak of the enhancement as one measure of enhancement, and integrates the enhancement-curve as another. Furthermore, it can create other graphs to help understand the data.

The programme designed to analyse multiples of these experiments can do all of the above, without creating the graphs that would be irrelevant in this instance. However, it can also compare multiple sets of data at different

concentrations and compensate for this. It, too, has a selection of plotting features that pertain specifically to the sort of data it is designed to deal with.

The parameters and settings used in the spectrofluorometer for taking these measurements are summarised in Table 7.

Variable	Pre-Enhancement	During Enhancement	Post-Enhancement
Experiment type	Emission Spectra	Emission Spectra	Emission Spectra
Measurement type	Raw Channels	Raw Channels	Raw Channels
Max its	15	15	15
Excitation (Enhancement) wavelength (nm)	350	200-300	350
Emission wavelength (Measurement range) (nm)	380-550	380-550	380-550
Exposure time	-	270 secs (4:30 mins) per run. Total = 540 secs (9 min)	-

Table 7: Parameters for Emission Profile Experiment.

5.4.4 Effects of Polarisers

When measuring the fluorescence of our samples, much scattered light is also detected. This is particularly the case when using a simulant such as milk powder, because these samples contain large quantities of small particles. This experiment was carried out to investigate the possibility of using polarisers to selectively remove some of this scattered light to improve the signal.

Polarisers are very useful for selectively eliminating the scattered light because when light reflects or scatters off an object it becomes polarised, (see Figure 5.4). So by using polarisers we can preferentially remove the scattered light;

the fluorescence should occur in all directions, so this will not be preferentially removed.

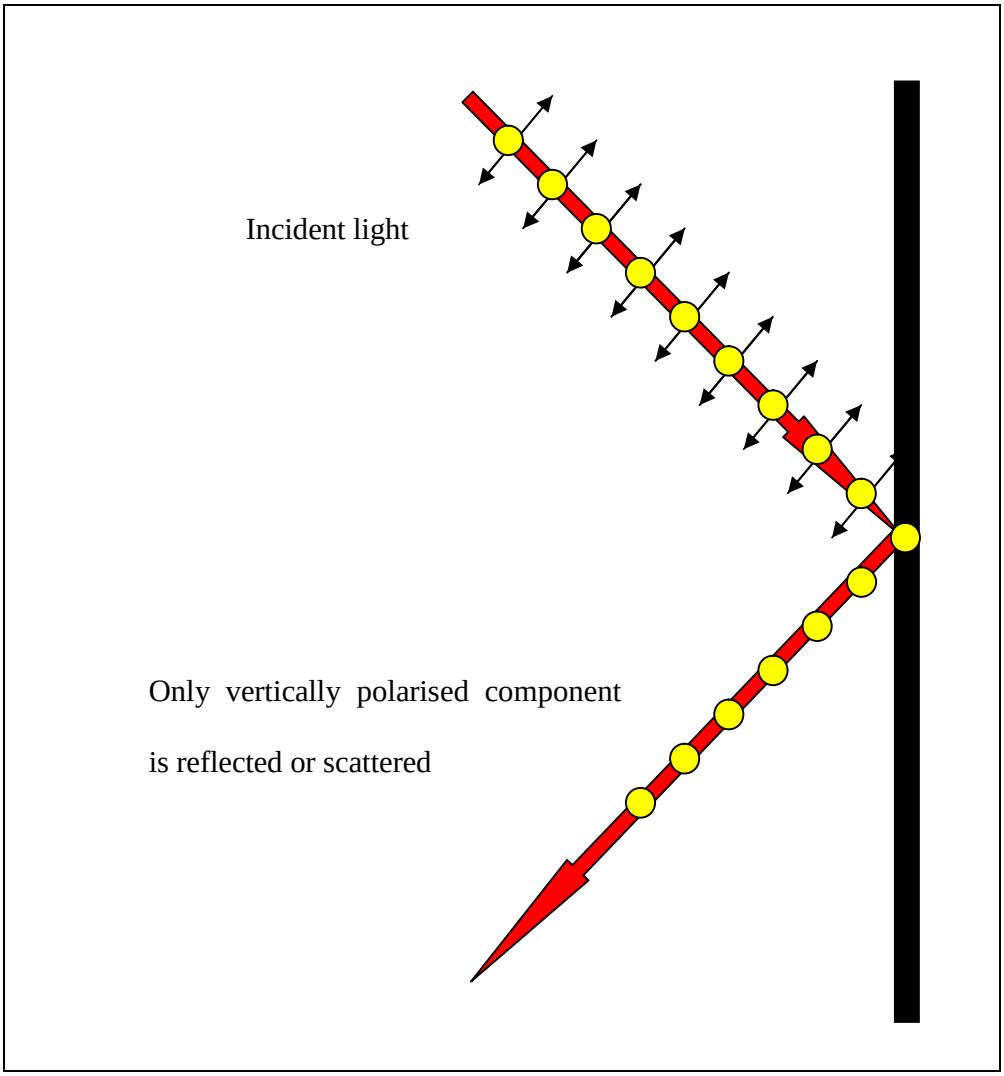

Figure 5.4: Polarisation by scattering or reflection.

We created a solution of 0.05 g of milk powder (Pam's) in 50 mL of water. We extracted 1 mL of this solution and mixed it in a quartz cuvette with 2 mL

of water, giving us 0.001 g of powder in the 3 mL solution. We enhanced this sample for 30 minutes using the water-cooled arc lamp set to 150 Watts.

After enhancement we placed the cuvette into the SLM 8000C Photon-Counting Spectrofluorometer. The parameters used to take the measurements are summarised in Table 8.

Parameter	**Value**
Excitation wavelength	340 nm
Measurement range	360-670 nm
Number of Increments	620
Integration time	2 seconds

Table 8: Measurement parameters used with the spectrofluorometer.

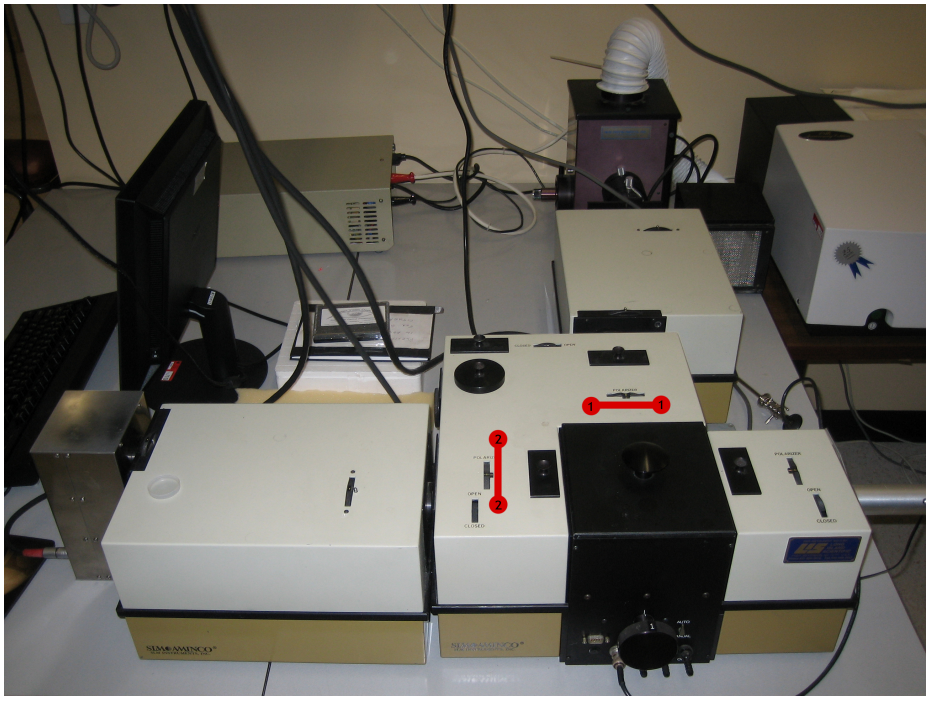

Figure 5.5: The spectrofluorometer, showing the location of the polariser-slots.

We used BVO UV dichroic polarisers (Bolder Vision Optik). The first polariser precedes the sample chamber; the second follows it. Figure 5.5 shows the machine with the location of the polariser-slots.

First, we took measurements using no polarisers, and then repeated the same measurements using a selection of polariser configurations, as shown on Table 9.

Scan #	**Polariser 1**	**Polariser 2**
1	None	None
2	Vertical	Vertical
3	Vertical	Horizontal
4	Horizontal	Vertical
5	Horizontal	Horizontal
6	Horizontal	None
7	Vertical	None
8	None	Vertical
9	None	Horizontal

Table 9: Polariser configurations used in this experiment.

5.4.5 Linearity of Response to Enhancement Time

When taking measurements of the CaDPA, it is important to know that the measurements are giving a linear response; this enables a relationship to be established between the amount of signal, enhancement and amount of CaDPA present. We measured the linearity of this response by creating a

sample of CaDPA in water. We measured the absorption spectrum to determine its concentration, ensuring it was low enough to transmit sufficient light. Using this solution, we created a new 1 mL sample before each measurement was taken. We put the sample into a quartz cuvette and placed it in the ISS PCI Photon Counting Spectrofluorometer. First, the pre-enhancement fluorescence was measured, before the sample was enhanced with 270 nm light. This was done by setting the monochromator to 270 nm and opening the shutters. After the desired enhancement time had elapsed, the post-enhancement fluorescence was measured. The parameters used for these measurements are summarised on Table 10.

Parameter	**Value/Type**
Experiment type	Spectra, Emission
Measurement type	Raw channels
Maximum iterations	15
Excitation wavelength	350 nm
Emission wavelength range	380-550 nm
Enhancement wavelength	270 nm
Excitation monochromator slits	2 mm
Emission monochromator slits	1 mm
Enhancement time	0-14 minutes

Table 10: Parameters used during pre- and post-enhancement measurements.

We repeated the measurements, so each enhancement time was done twice, and calculated the mean enhancement from this.

6 Results and Discussions

In this chapter we present the results of our experiments and show the data analyses we used to interpret them. Some of the earlier experiments presented here were performed to ensure the accuracy of the later ones. More specifically, we had to ensure that our measurement technique did not interfere or alter the fluorescence of the samples we were measuring. Also, we had to ensure that the machine was performing properly and giving consistent measurements and readings. Once this had been established, we could continue and perform our main experiments.

6.1 Checking Null Effect of 350 nm Light

The Vinci software controlling the spectrofluorometer outputs all data in .ifx files. These were converted to .txt files using the batch processing command “ren *.ifx *.txt” in Command Prompt. The data was then analysed with the plotaRaph programme written as part of this thesis, the full code of which can be found in Appendix 1. This programme is a very useful generic plotting programme which can be readily adapted for use with different types of data.

This programme has a selection of user inputs which can also be hard-coded in. We generally used a combination of these user-inputted and hard-coded variables depending on which variables were being changed most frequently.

After collecting these values the software runs the DataExtrator function, which reads the data out from the text file, ignoring the rows of header, and saves the data into an array. This data array is then fed back into the main programme. From here the options of running the data smoother and data manipulator are presented. The smoothing programme calculates the mean value of a user-defined number of points from an array, and saves it to the central point. The smoothing size must be an odd number; if an even number is inputted it will automatically reduce this by one and inform the user of this change. The data manipulator can be set up to do any further manipulation of the data if this is required. The next thing it does is to find the maximum value of that dataset. This also has the option of restricting the search area to a certain range of the data. This feature was valuable in our case, as we were not interested in the scattered peaks that occur at either end. After the MaxFinder has found the maximum value it runs the plotter programme which plots the current dataset. This process is repeated until all the pre-specified datasets have been done.

After completing this, the programme runs the PlotDecorator which requests user input of all the labels and offers options of either automatic or user-defined axes and legends. The MaxValuePlotter plots the collected maximum

values against a user-defined array specified earlier, once again asking for user input for the graph labels.

Experiment number	Enhancement time (at 350 nm) (mins)
1	0
2	5
3	10
4	15
5	30
6	90
7	0

Table 11: Enhancement times used in this experiment.

The enhancement times used in this experiment are summarised in Table 11. The emission profile was measured from 360 nm to 700 nm. This starts slightly above the excitation wavelength and results in a very high intensity to be measured at the lower values. The other end corresponds to the second harmonic, and thus also has scattered light. These intensities are too high and of no use to us. We have only plotted the data from 370 nm to 650 nm to remove these distracting scattered peaks.

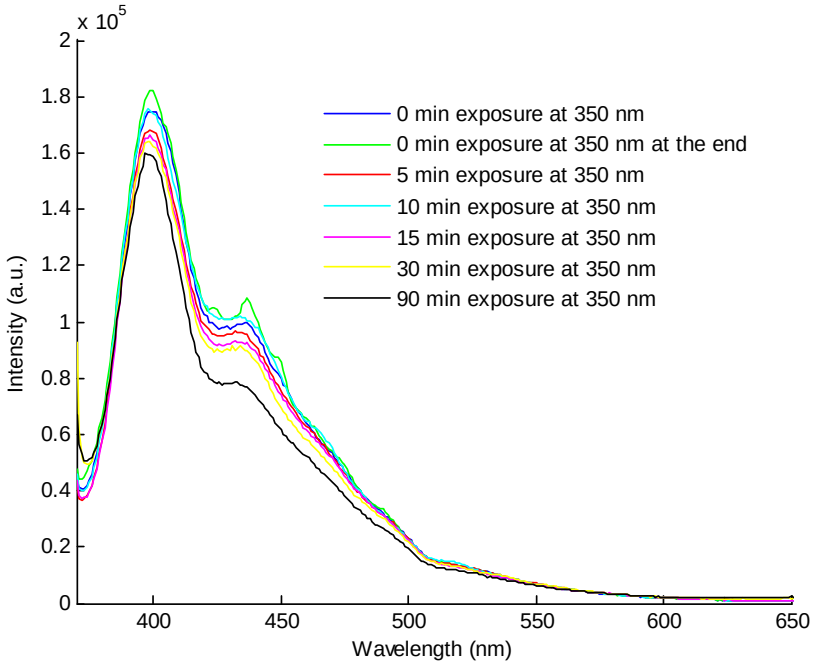

Figure 6.1: The effect of 350 nm exposure. The emission profiles after exposing the sample to 350 nm light for various durations. 350 nm excitation wavelength used.

The graph in Figure 6.1 shows the emission profiles measured after each enhancement by the 350 nm light. It is immediately obvious that there is no significant change in fluorescence occurring. Another way to look at it is to extract the peak values and plot these against enhancement time, as shown in Figure 6.2. Here it is clear that there is no actual change occurring in the peak values and the slight variation is simply due to noise in the data.

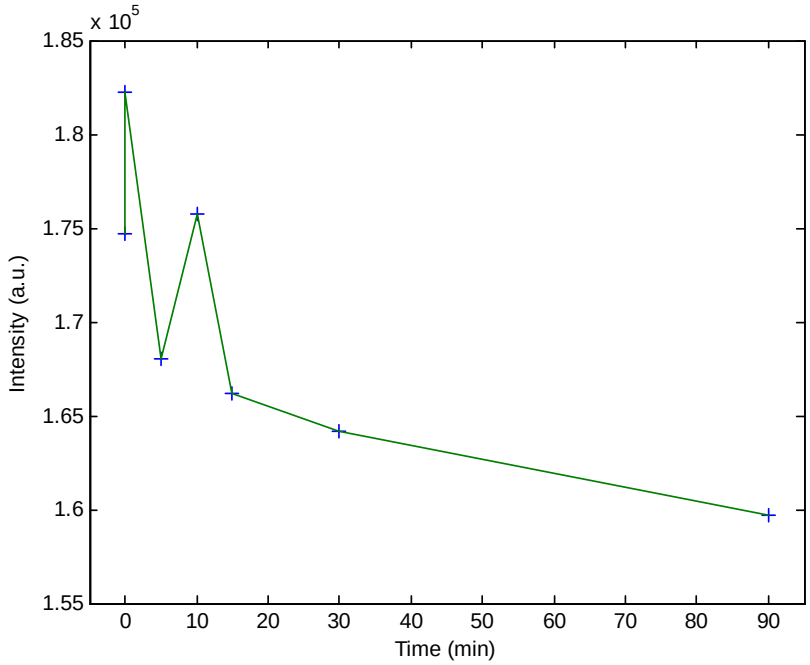

Figure 6.2: The peak values from each emission profile. Plot showing that no enhancement occurs because of exposure to 350 nm light.

Therefore our measurement technique is valid and there is no reason for concern about this having any effect on the final measured results.

6.2 Finding the Appropriate Warm-Up Times for the Apparatus

Our first measurements of the output gave a rather surprising result; a graph of this data can be seen in Figure 6.3. This clearly shows that after an initial increase in intensity, it decreases drastically over a long period of time. The initial increase occurs during the first 600 seconds, (10 minutes), followed by a long decrease through to about 15,000 seconds, (4 hours and 10 minutes).

Following this, the machine appears to have warmed up and stabilised, because the output is fairly constant. The percentage change in the intensity can easily be calculated using the following:

$$\text{Percentage change} = \%\Delta = \frac{\Delta I}{I}$$

Where
ΔI is the change in intensity
I is the intensity after warm-up

Using this equation, we found the percentage change occurring during this warm-up time was 36% over a period of 15,000 seconds. From this, it was easy to conclude what the warm-up time was and that it was important to allow the apparatus to warm up before using it. However, we wanted to find out which components were causing this warm up so we could be specific in what we warmed up. Because the intensity levelled out after 15,000 seconds, we decided to run the follow-up experiments only for this length of time.

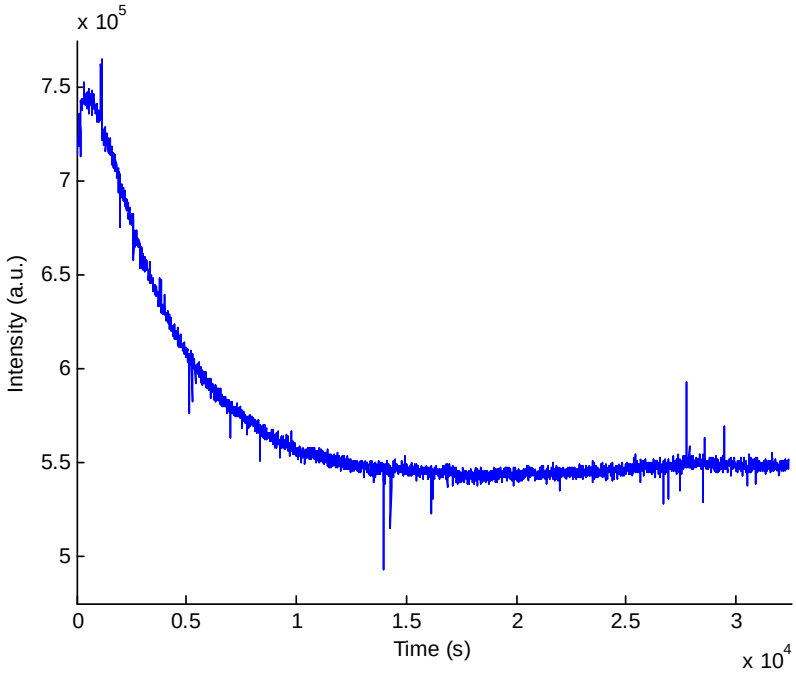

Figure 6.3: Change in measured intensity as apparatus warms up. This shows a 36% change in intensity during the first 15,000 seconds.

We set up these experiments so that some pieces of apparatus were already warmed up while others were cold. We used a selection of combinations to determine which of the components was actually contributing, and then how much each of these contributions was.

Our first aim was to determine whether or not the computer was in any way having an effect. We had noticed some unusual changes depending on how the software was left, i.e. if it was waiting for a response from the user or not. First, we checked and verified that the computer itself was not contributing. We performed two experiments: for the first, both the lamp and fluorometer

were on, while for the second, the computer was also on. The excitation-channel graphs are shown in Figure 6.4 and Figure 6.5

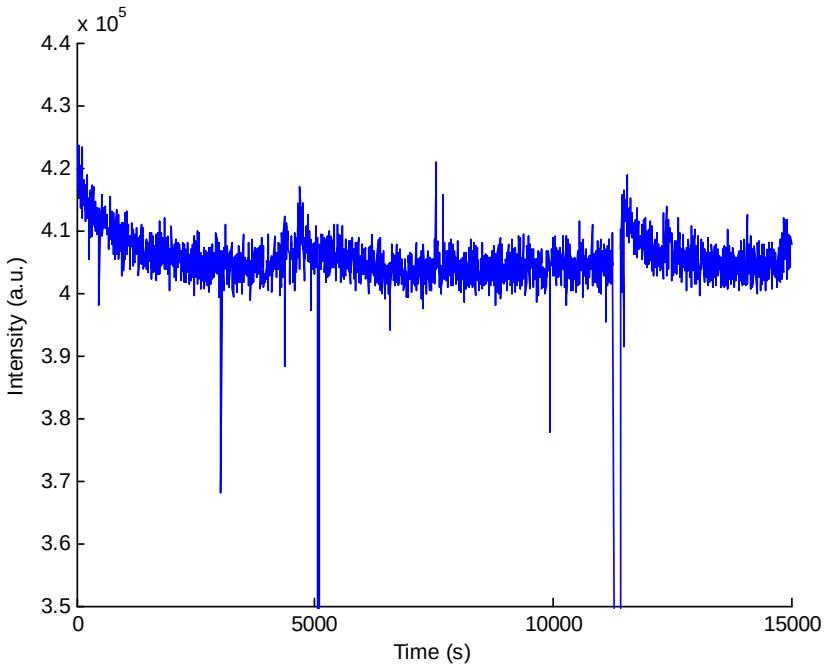

Figure 6.4: Excitation-channel graph; fluorometer and lamp warmed up, computer off.

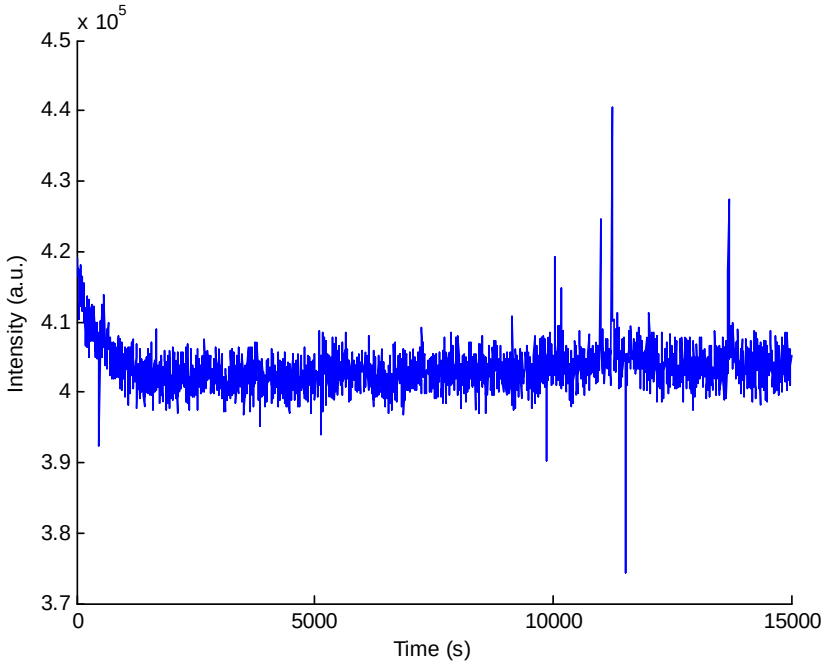

Figure 6.5: Excitation-channel graph with all components warmed up. As above but with the computer also on, showing no significant change.

From these graphs it is apparent that the computer has no contributing effect. Both have comparable intensity changes, (2.7% for Figure 6.4; 3.0% for Figure 6.5), and times of approximately 1,400 seconds (~23 minutes). So we can safely conclude that the computer itself was not having any effect. However, the same minor change apparent in the previous experiment was still evident. This change of about 3% did not appear when we ran a further experiment which was started sometime after the first had finished but the computer was waiting for user input as to whether or not the data from the first was to be saved. Figure 6.6 shows this graph, with no apparent warm-up effect.

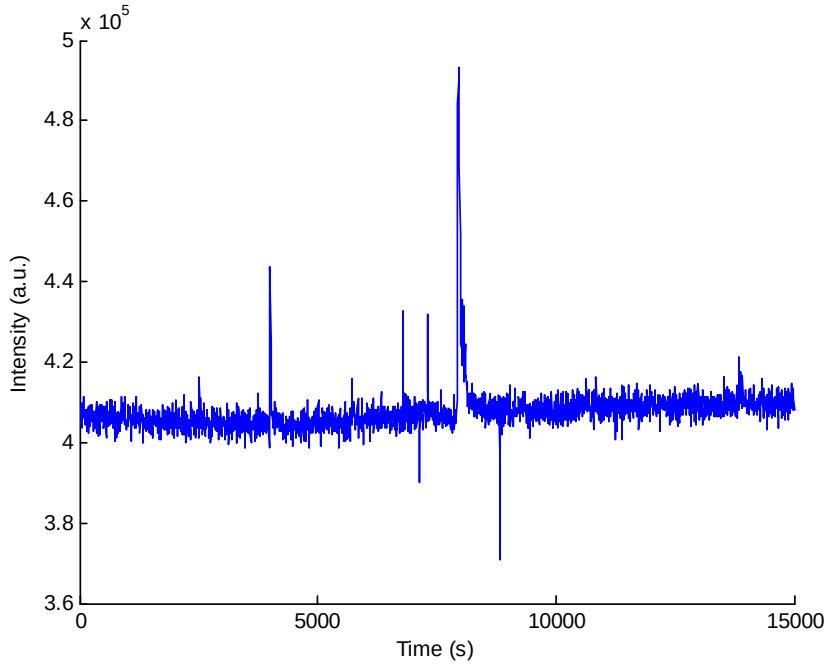

Figure 6.6: Excitation channel with all apparatus on and the shutters open. Graph showing intensity for experiment begun immediately after waiting for user input with all components warmed up.

After looking at what was happening in the fluorometer while it was waiting for user input we found that it left the shutters open. Light was still entering the PMTs, so we concluded that the computer had no effect, but the shutters needed to be open to remove this ~3% variation. So, having understood this part, we moved on to the lamp and fluorometer itself.

The warm-up time for the fluorometer is shown on Figure 6.7. This shows the warm-up taking the entire time displayed on the graph. The % change was calculated as above and found to be 26.4%.

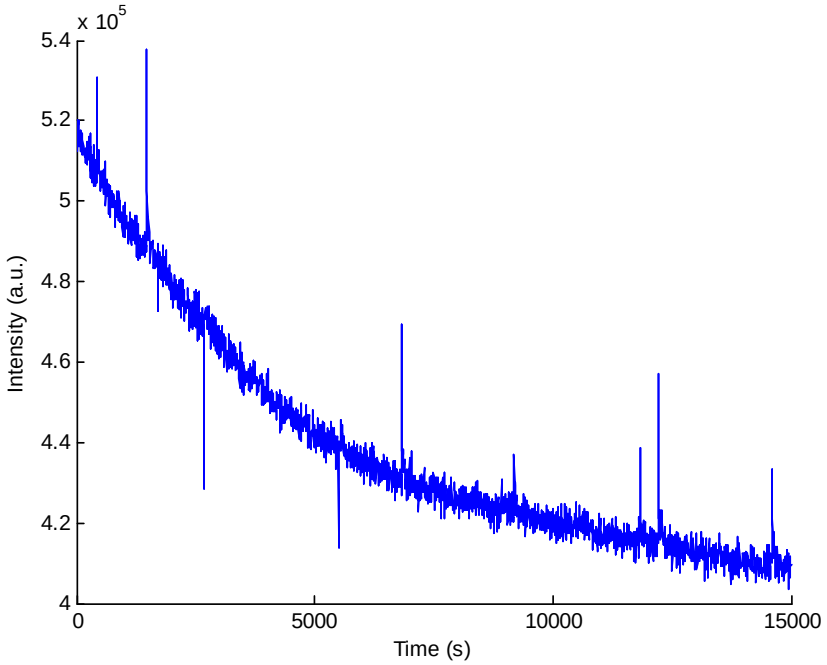

Figure 6.7: Excitation channel; fluorometer and computer off, lamp warmed up. Showing the warm-up time for the fluorometer.

We set up an experiment where we warmed up the fluorometer and computer. We then turned on the lamp and began taking measurements immediately. Figure 6.8 shows the results. It is interesting to note that this graph shows an initial increase before decreasing, which did not occur with other components but was observed in our initial experiment. This may be because of some thermal effects in the lamp. The warm-up time for the lamp appears to be about 15,000 seconds (~4 hours) with a % intensity change of 10.4%

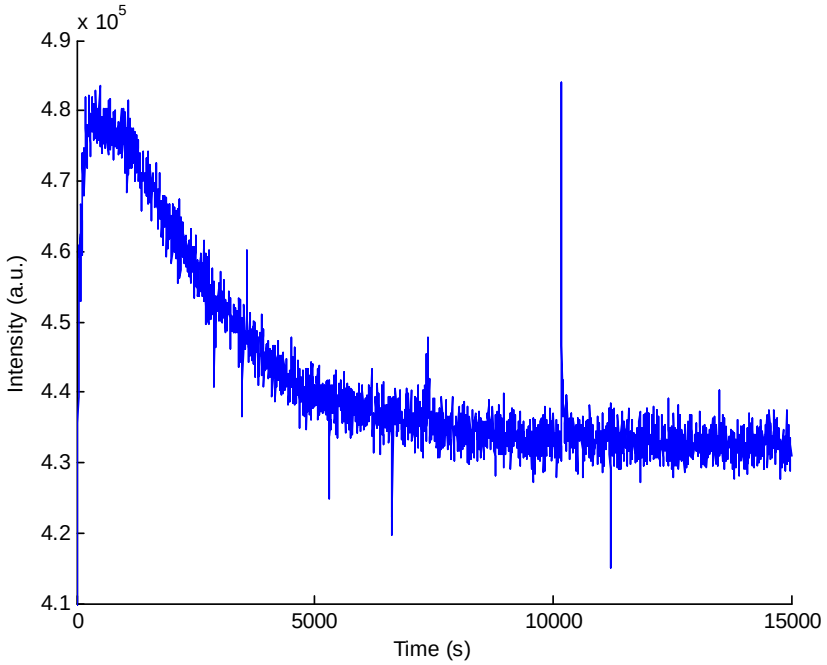

Figure 6.8: Excitation channel; lamp and computer off, fluorometer warmed up. Showing the warm-up time for the lamp.

Component	% change in intensity during warm-up	Warm-up time (seconds)
Lamp	10.4	8,000
Fluorometer	26.4	15,000
PMT exposure to light	3.0	1,400
Total from initial experiment.	36.3	15,000

Table 12: Summary of warm-up contributions in the excitation channel

It is interesting to note that the initial % change was 36.3%, and the sum of the changes which occurred from the lamp and the fluorometer is 36.8%. This shows that the two components have a cumulative effect. The other effect, caused by exposure of the PMTs to the lamps, occurs over a much shorter

time-frame and thus has very little influence overall, due to it having decreased to less than half by the time the lamp reaches its peak. The changes from the lamp and flourometer are much greater, and occur over a significantly longer time-frame than the effects of the exposure of the PMTs, so the latter effect is negligible.

The results discussed so far have all been for the excitation channel. The reason why these have been discussed in such detail is that they show a much greater change, and are thus more important. The emission channel graphs are all fairly similar; they have much shorter warm-up times of typically around 1,600 seconds, tending to increase gradually and then flatten out rather than having the long decreasing period, (in the middle), of the excitation channel. A typical graph of emission channel warm-up is shown in Figure 6.9. We found no conclusive reasons as to why this occurs; however, our aim in this experiment was to determine how long we had to warm up the apparatus, and then more specifically which components needed warming up, and both of these were ascertained.

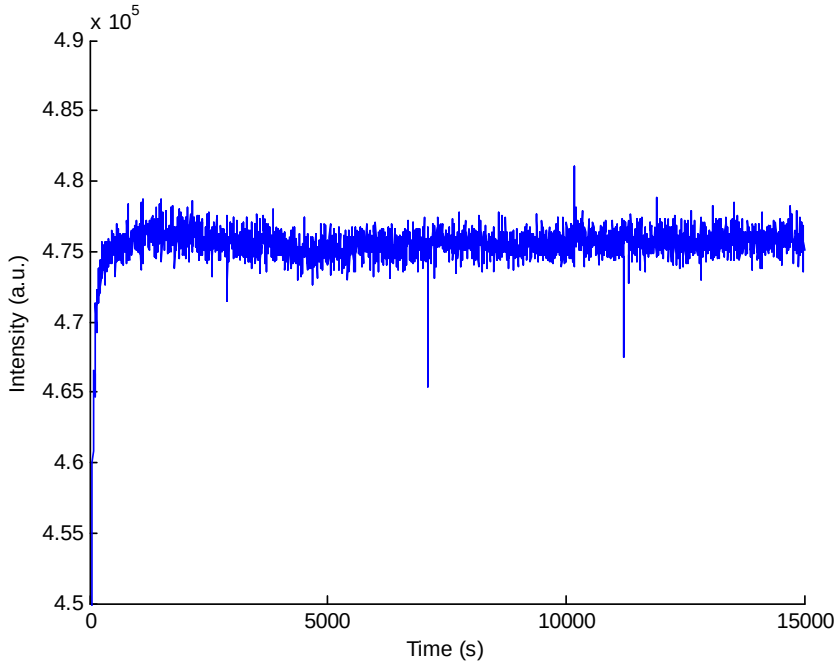

Figure 6.9: Emission channel; lamp and computer off, and fluorometer on. Example of a typical emission-channel warm-up graph.

All analyses and the plotting of the graphs for this experiment were done using the plotaRaph programme mentioned previously. The nature of this programme allows a diverse range of data types to be analysed.

We found that the warm-up time for the apparatus had to be at least four hours to ensure that stable and reliable results could be obtained. Furthermore, we found that all components needed warming up, most importantly the lamp and PMTs. Having the computer running with the light shining into the PMTs did have a slight effect, but this was over a much shorter time period of about 23 minutes, so this could occur while other aspects of the experiment were

prepared. The computer being on actually made no real difference, it is simply the easiest way to control the shutters and it did not matter if it was on anyway. So our future experiments always had everything running to warm up for at least four hours to ensure that consistent results were obtained.

6.3 The Emission Profile of CaDPA and its Variation with Concentration

This series of experiments was performed to determine the nature of, and quantify, the change in fluorescence that occurs in CaDPA when it is enhanced by light. It is obvious that a change does occur, as demonstrated in Figure 6.10; a graph of the fluorescence of CaDPA both before and after enhancement by light.

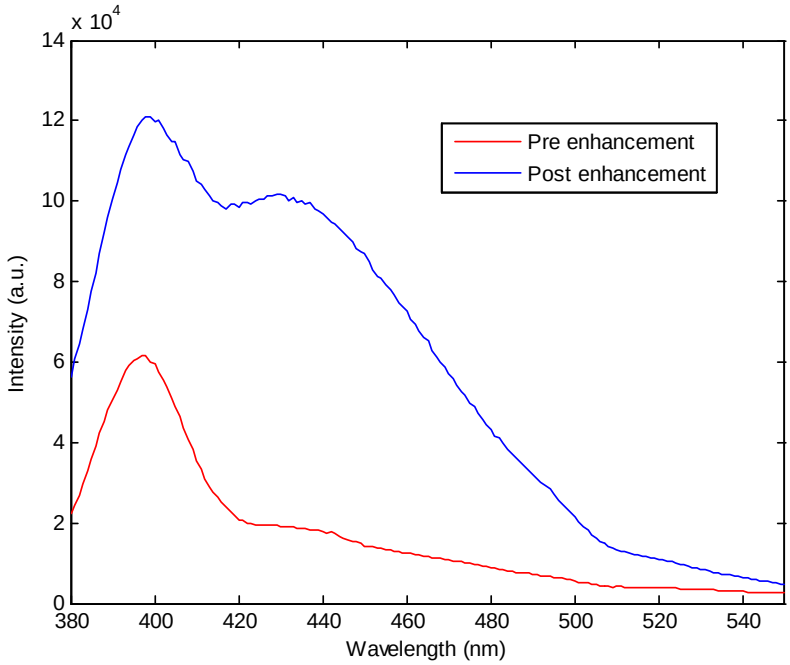

Figure 6.10: Fluorescence profile of CaDPA before and after enhancement by 230 nm light. Excitation wavelength was 350 nm.

If measurements are taken too near the wavelength of the excitation light, that is, the light being used to create the fluorescence, a large peak of scattered light is detectable, restricting measurement of the enhancement. Measurements can only be taken at wavelengths greater than the excitation wavelength. We used 380 nm to minimise scattered light. A further restriction is naturally imposed at the upper limit, caused by the second harmonic of the excitation light wavelength; this occurs at twice the excitation wavelength. We used 550 nm as our upper limit because we found that the majority of the enhancement occurred below this; there was no need to collect unnecessary data and this reduced the time it took to perform the experiments.

The change in fluorescence profile can be seen when the enhancement is done by a range of wavelengths so we performed the following set of experiments to determine what the optimal wavelength to use for this is.

We enhanced the sample using light from 200 nm to 300 nm increasing in increments of 10 nm. We measured both the pre- and post-enhancement profile and recorded the intensity of the light being used to enhance the sample. Our analysis programme was then used to create an emission profile, as demonstrated by the sample in Figure 6.11.

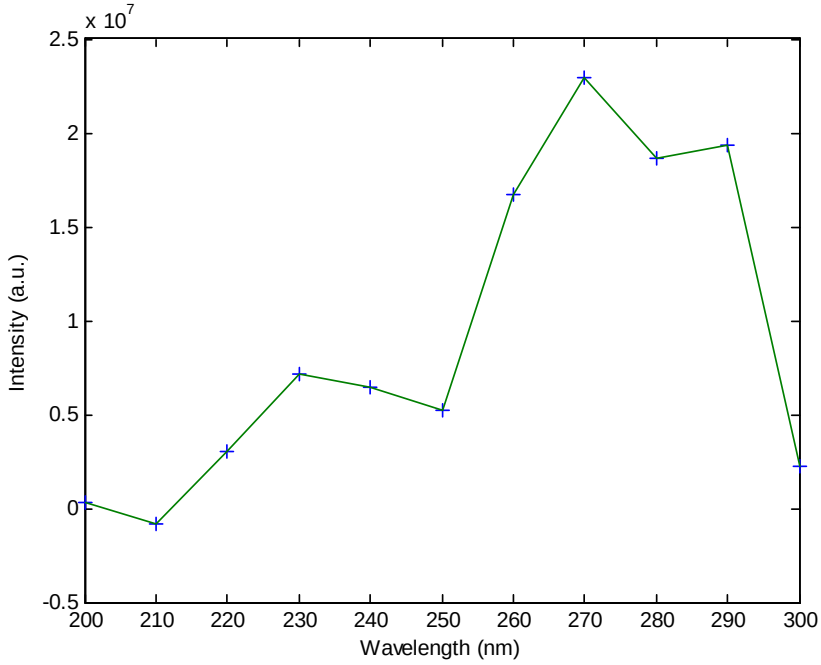

Figure 6.11: Non-normalised emission profile of CaDPA, from the integral of the fluorescence enhancement data. Excitation wavelength 350 nm, not normalised for variations in enhancement light intensity.

The lamp we used for this enhancement does not have a linear Intensity profile, see Figure 6.12. So we had to normalise the emission profiles we collected for this variation, which resulted in a very different emission profile. Figure 6.13 shows an emission profile that has been normalised for variations in the enhancement lamp intensity.

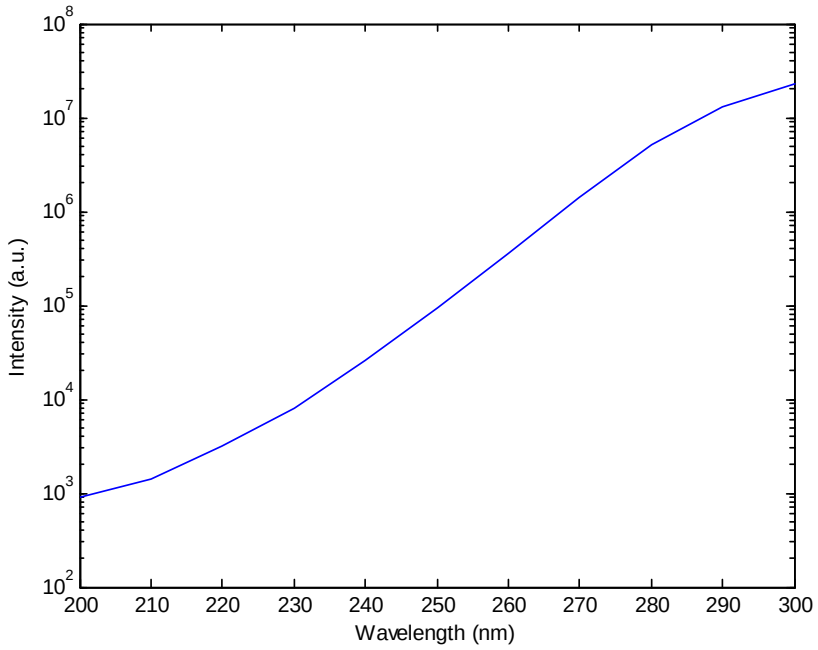

Figure 6.12: The enhancement lamp intensity profile.

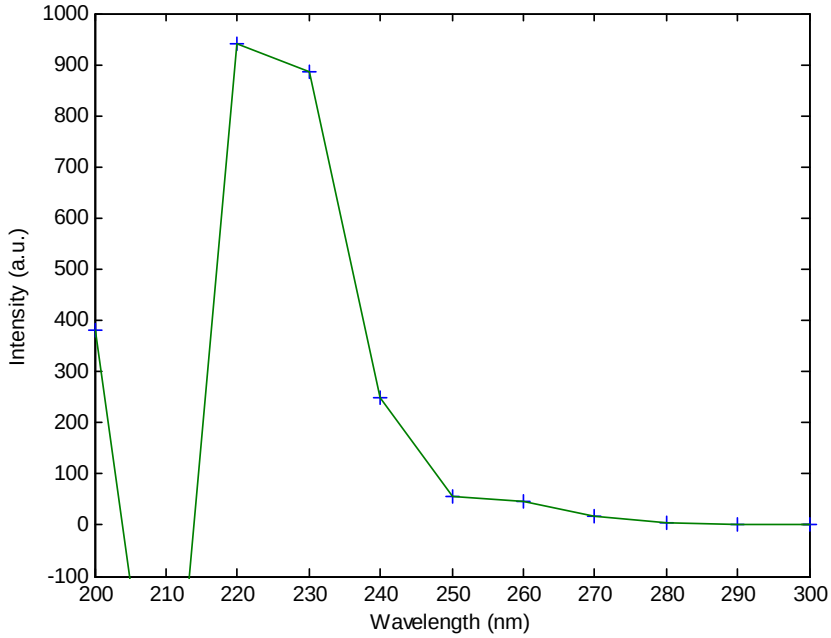

Figure 6.13: Normalised emission profile of CaDPA, from the integral of the fluorescence enhancement data. Normalised for the variation in enhancement lamp intensity. 210 nm value is negative because of a spike in the pre-enhancement data.

Our early experiments showed some variation in our results, so we investigated the effects of concentration. These results are discussed in the following section.

6.3.1 Emission Profile of CaDPA with Concentration Normalisation

We measured the emission profile of CaDPA, and repeated this 4 times using solutions that differed in concentration. This allowed us to find possible variations that might have occurred due to the variation of this parameter.

The sample was created and an absorption spectrum was measured, using the GBC UV-Visible Cintra 40 Spectrometer. The sample spectrum in Figure 6.14 clearly shows that this is CaDPA because there is a triple peak, with a central peak at 269 nm; pure DPA would have a single peak.

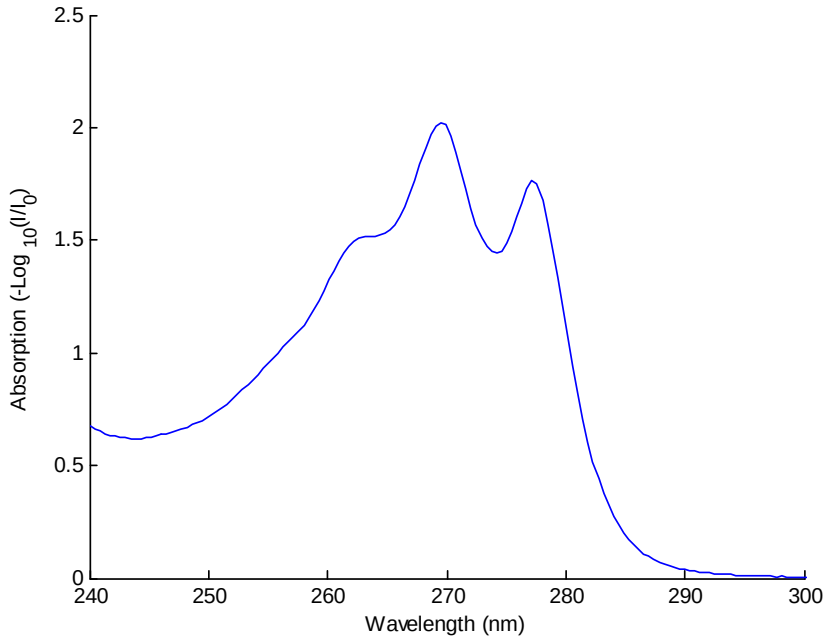

Figure 6.14: Absorption Profile of CaDPA. Sample absorption spectrum of CaDPA showing triple peak centred at 269 nm.

This spectrum was measured for each of the samples used in this experiment, and the absorption, measured at the central peak, was used to calculate the corresponding concentration of the sample. This is summarised on Table 13.

Expt #	Absorption (measured)	Concentration (calculated) (µMol/L)
1	1.31832	9.553
2	0.09148	0.6629
3	2.3125	16.757
4	0.0547976	0.3971

Table 13: Absorption and corresponding concentration of all experiments in this series.

6.3.2 Data Acquisition and Analysis

Vinci (Version 1.6 SP4), the software running the Spectrofluorometer, outputs its data as .ifx and .CHD files. We batch-converted the file names to change the ifx files into .txt files using the command “ren *.ifx *.txt” in Command Prompt. This makes them ready for use with the programmes we wrote in MATLAB (The Mathworks Inc, Natick, Massachusetts, USA), to analyse the data from these experiments.

6.3.3 Analysis Programmes We Wrote and Used for this Experiment

To successfully interpret and analyse the data from these experiments, and draw clear results from it, we created our own analysis programmes. These are described in the following two sections. The full MATLAB code for them can be found in appendices 2 and 3.

6.3.3.1 Single-Concentration Emission-Profile Analyser Programme

This first programme was designed to analyse datasets from single-concentration experiments and create emission profiles of these. However, it was also designed to analyse other data. It can select various aspects of the data and process it in different ways. Thus it has options to hard-code in all variables or to use the user-input functions. We usually used a combination of the two, depending on which variables were frequently changed. The selection is done by “commenting out” those lines which are not needed.

After all required variables have been set, including those input by the user, the programme calls a separate function which reads the data from the text files. The names of these, and the legend labels, come from text files in the same directory. When reading the data, it ignores rows of headers and stores the data into an array. This array is then fed back to the main programme for use. It repeats this for all four sets of data (pre-enhancement, the two collected whilst enhancement was taking place, and post-enhancement). The necessary data are then sent through the data manipulator, which can be set up to further process it as required; usually, we simply used it to divide the emission column by the excitation one, if this was desired. Following this, a separate function can be called which smoothes the data. This takes in parameters for the smoothing size and averages the data over the number of points specified.

If an even number of points is chosen it automatically changes it to one less and outputs a warning message to the user to inform them that this has been done.

Next, a separate array is created with the enhancement values; this stores the wavelength into column 1, the enhancement intensity into column 2, and, if selected, the normalised enhancement intensity into column 3. The enhancement is calculated by subtracting the initial values from the post-enhancement value for each entry in the array. The normalisation divides the enhancement by the integral of the total enhancement light from both enhancement procedures.

Another function finds the peak value of the enhancement curve, which can be restricted to a certain domain to avoid the scattered tails that appear near the excitation wavelength, as was the case in some of our experiments. It also integrates the enhancement and stores this into a separate array. All data are stored into a large cell array in case further analysis is to be done manually afterwards. Once all the data for a particular wavelength have been collected, it plots the appropriate type of graph from one of the six options given: Pre-enhancement values, post-enhancement values, enhancement, or any of these options but normalised for variations in the enhancement-lamp intensity. It also checks to ensure that no impossible combination has been chosen, such as

plotting calibrated values if normalisation has not been done. Plotting is done using yet another separate function which plots the data, automatically assigning a different combination of line type and line colour to each series. Once this is completed, the whole process is repeated for the other datasets.

After the data have all been collected, stored, and plotted, the programme calls the plot decorator function, which automatically adds the title, legend, and axis labels corresponding to the graph. This is automatically created by extracting the names from a separate text file.

Once the main graph has been created, it plots all the others that have been selected. Several options are available, including the pre- and post-enhancement values as a separate plot for each enhancement wavelength, the peak values for each enhancement wavelength, and the integrated enhancement values for each wavelength. All these graphs also automatically receive appropriate titles, legends, (where appropriate), and axis labels.

This programme was used to analyse the data we collected whilst performing the single concentration experiments; when analysing data from multiple experiments, and comparing these, we required a more specialised programme as described below.

6.3.3.2 Multi-Concentration Emission-Profile Data Analyser

This programme is able to analyse multiple datasets and compare, normalise, and average them. This enables many emission profiles, taken at different concentrations, to be compared.

Input of the measured absorption values corresponding to the experiments, is required by this programme. It also requires the name of one text file for each concentration which contains the necessary information that allows the computer to find all other required files. Once again, all data must be converted to .txt files before they can be used.

After calculating the concentrations of all the solutions used for taking the measurements, it collects the data by extracting the relevant file-names from the inputted experiment list and calling the function MultiDataFunc.m. This is based on the single-concentration analyser, converted to a function and with all unnecessary parts removed. This extracts all the data and performs all required steps to send the data back to the main programme. All of the data is then stored into a large cell array for later use. It also creates a few smaller arrays for convenience.

After collecting all the required data and storing it, it starts to perform the required analysis and plotting of graphs. First, it creates a plot of the data before any further data analysis or normalisation takes place.

It begins by creating a series of graphs comparing the emission profiles for the different concentrations, using both peak values and integrated intensities, and normalising these for different combinations of concentration and enhancement-lamp intensity variation.

It then averages together the four different experiments on each of these graphs and creates four graphs showing the mean values of these. To ensure that the graphs are consistent, it takes the mean graphs that had not previously been normalised for variations in enhancement-lamp intensity, and normalises them thus. This is done by dividing the mean values by the summation of the enhancement-lamp intensities used, which are systematically extracted from the data.

The intensity for each concentration at a given wavelength is extracted, and each wavelength's response is plotted on a graph of concentration and intensity. The mean value at each concentration is calculated and sent, together with associated uncertainty values, to a new plotting programme, SinglePlotter.m, which plots these. This also calculates a least-squares fit and plots this onto the same graph. It then normalises the mean values at each concentration for the corresponding concentration, and re-plots it using the same plotting programme. All these are also re-plotted on a semi-log scale axis to make the lower concentrations more visible and clear.

The programme automatically labels all axes and gives each graph a suitable title. Where appropriate, a legend is also automatically added.

6.3.4 Results

The first way to look at the data is to compare the emission profiles from each of the different concentration graphs. There are two ways of finding these; the area under the enhancement curve can be integrated, or the peak values are found when either of these is plotted against the wavelength used to enhance it. This is called an enhancement profile. Both were found to result in similar-looking graphs, clearly with different axes, but very similar emission-profile shapes. We will only be showing our results for the integrated values to avoid repetition. Figure 6.15 shows the emission profiles for each concentration before any normalisation has been done.

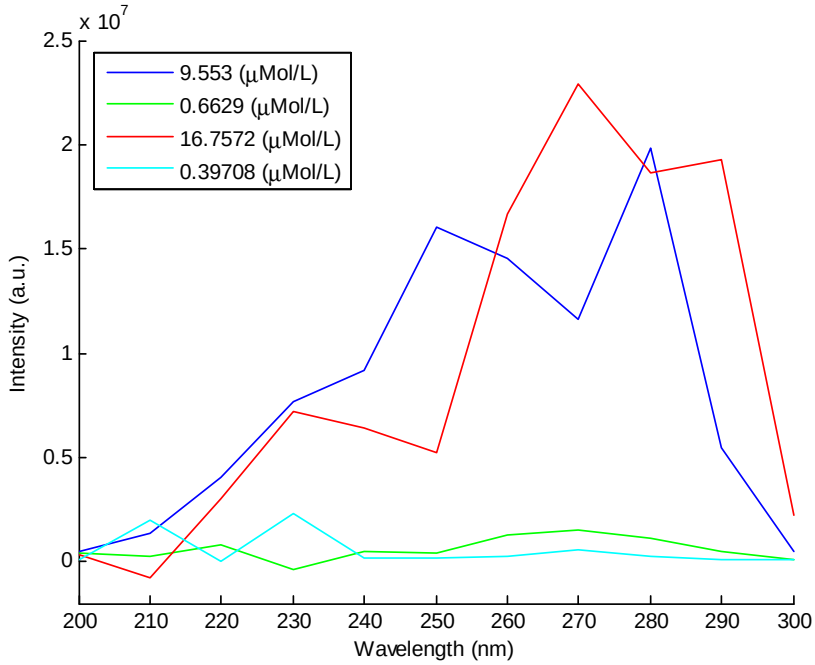

Figure 6.15: Emission profile from integrated values, not normalised for variations in concentrations or enhancement light intensity. All values calculated by integrating the enhancement. Excitation wavelength 350 nm.

This gives a rather distorted view because there are two factors which are not equal for all values. These are the concentration, and the variation in the enhancement-lamp intensity. This first graph gives the impression that longer wavelengths of around 270-280 nm are producing most of the enhancement.

Normalisation for concentration can easily be achieved by dividing each dataset by the corresponding concentration of the solution used to make that measurement. Doing this produces the graph shown in Figure 6.16.

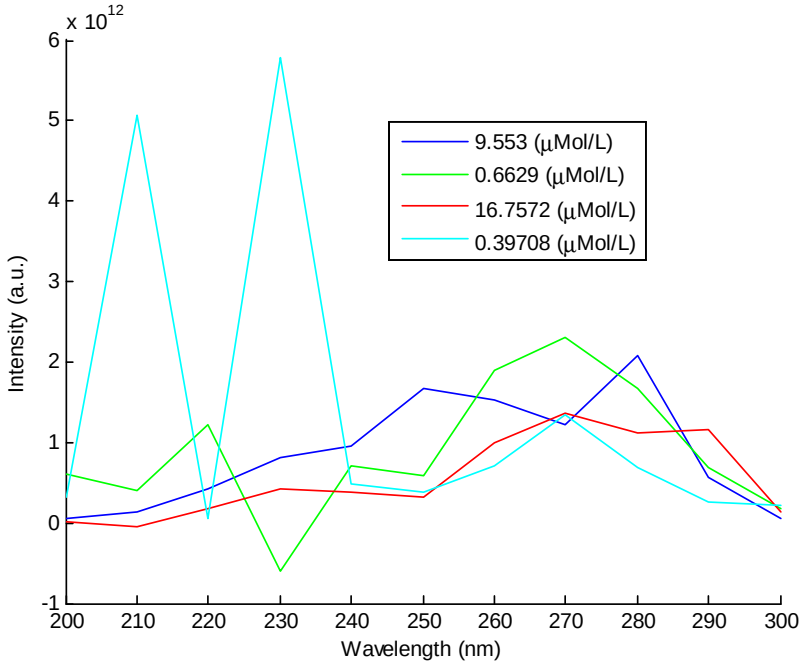

Figure 6.16: Emission profile from integrated values, normalised for concentration only. All values calculated by integrating the enhancement. Excitation wavelength 350 nm.

This obviously puts much greater emphasis on the low-concentration samples, and makes the four emission profiles look rather similar, except for the significant spikes at very low wavelengths on the lowest concentration profile. The graph still suggests that the optimal enhancement wavelength for most concentrations is 270-280 nm. So, having brought the profiles into agreement, we needed to correct them once again to compensate for the nature of the enhancement-lamp intensity profile (see Figure 6.12).

The emission profile in Figure 6.17 demonstrates the importance of the final normalisation. The emission profile now shows a strong peak at low

wavelengths and very little at long wavelengths. This was expected because of the significant variation in the enhancement lamp.

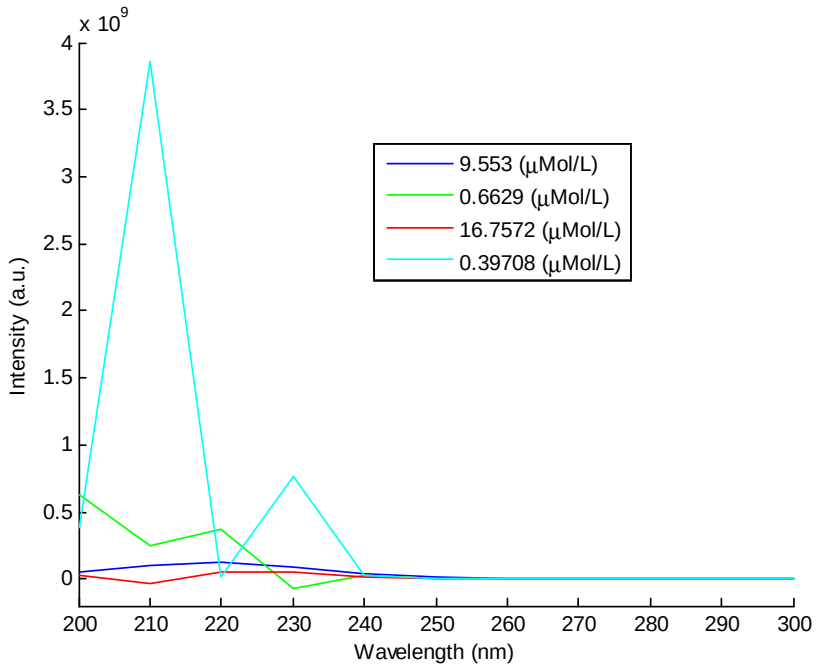

Figure 6.17: Emission profile from integrated values, normalised for variations in enhancement lamp intensity and concentration. All values calculated by integrating the enhancement. Excitation wavelength 350 nm.

The effect of the intensity profile of the enhancement lamp on the emission profile of CaDPA was expected, given our measurements. However, we expected this to be flatter, with less variation over the range of wavelengths in question. We have checked and taken these measurements many times; they have been in agreement and we cannot find any reason why they should be deemed incorrect.

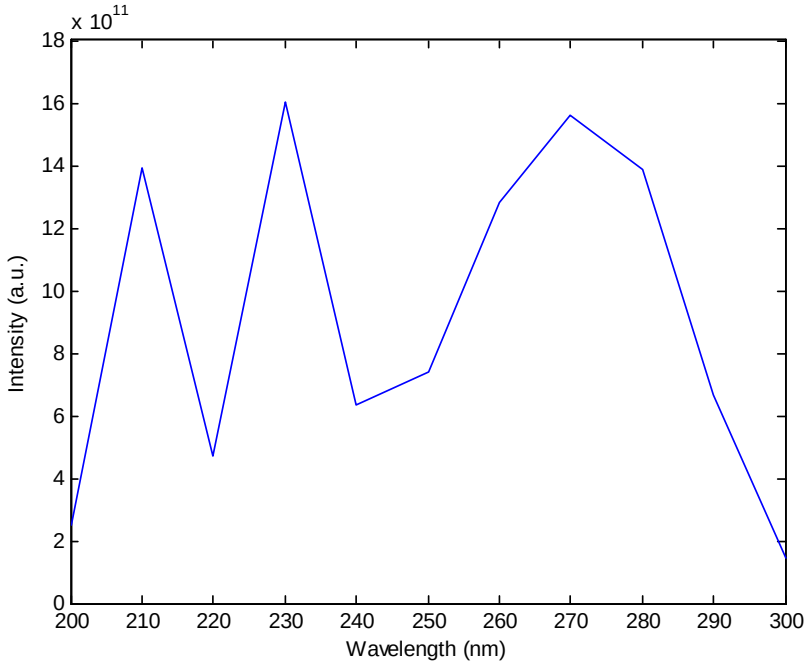

Figure 6.18: Mean of the integrated values normalised for concentration only. Emission profile of CaDPA calculated by taking the mean values at each concentration; only normalised for concentration, not variation in enhancement-lamp intensity. Excitation wavelength 350 nm.

The true emission profile can be created by taking the averages from all concentrations. The data normalised only for variation of concentration is shown in Figure 6.18. This shows a profile with a comparatively wide peak centred at 270 nm, and also the two peaks from the lowest concentration emission profile at 210 nm and 230 nm. The final emission profile, calculated by averaging the emission profiles from all the concentrations, which have already been normalised for concentration, and intensity profile of the enhancing lamp is shown in Figure 6.19. This profile has very little definition

above 250 nm and is strongly dominated by the peaks at 210 and 230 nm from our lowest-concentration experiment.

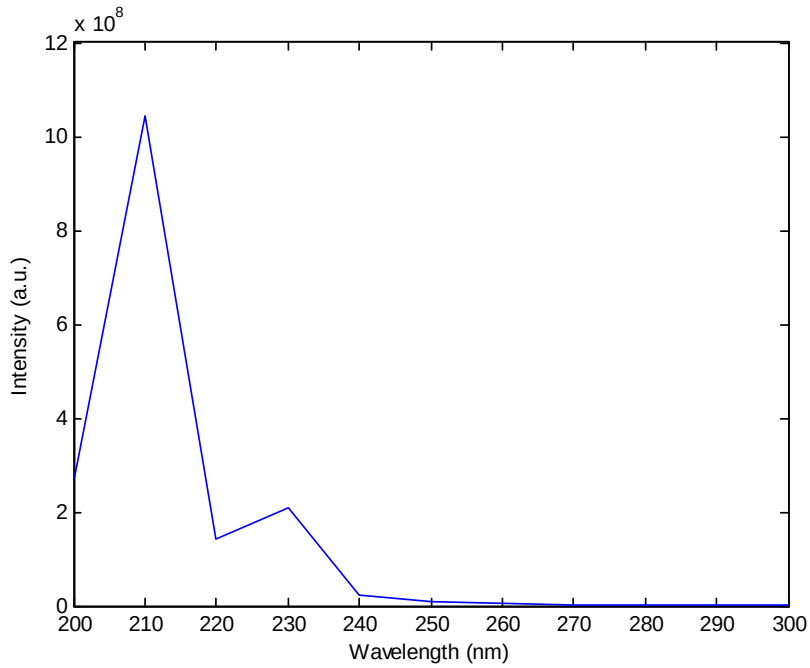

Figure 6.19: Mean of the integrated values normalised for variations in enhancement-lamp intensity and concentration. Emission profile of CaDPA, found by averaging the emission profiles at each concentration which had been normalised for concentration and enhancement-lamp intensity. Excitation wavelength 350 nm.

So the ideal wavelength to use when enhancing CaDPA is 210 nm. However, as most light sources are more intense at longer wavelengths, it may be more effective to use the 270 nm peak.

We extracted the data pertaining to the intensity of the enhancement at each concentration. These values were averaged to find how the mean enhancement changed with the concentration. This is shown in Figure 6.20; a fairly linear

response with increasing concentrations. This is to be expected at low concentrations, where only a small percentage of the light entering the sample is absorbed. We also performed an experiment at much higher concentration, not shown because we were unable to determine the exact concentration of the sample; the absorption was greater than the capabilities of our machines allowed us to measure. This higher concentration point has a lower intensity than the two highest concentrations shown here. We suspect this was the case because the solution was too absorbent, so that much of the light was not passing through the sample and thus either not causing enhancement, or re-absorbing the fluorescence.

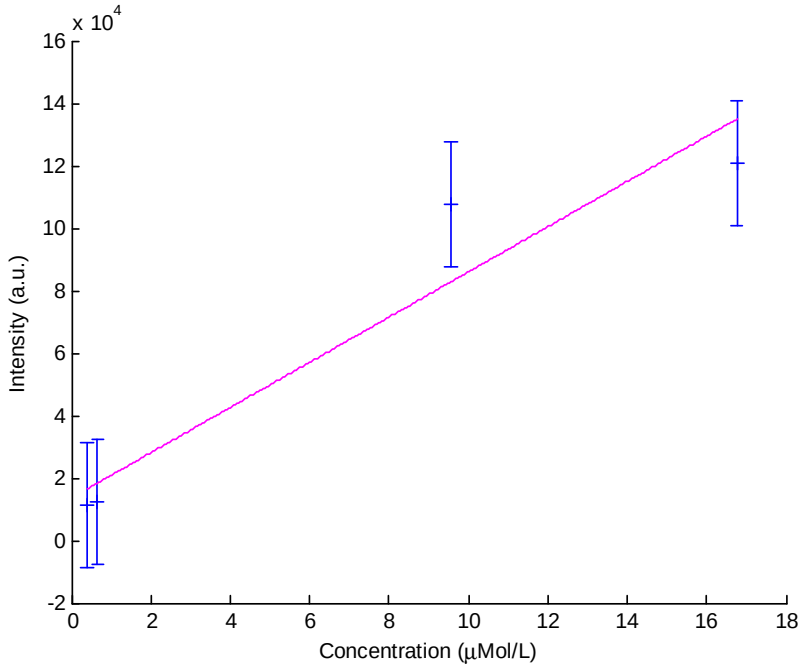

Figure 6.20: Mean intensity from each wavelength at a given concentration. The average enhancement occurring at each concentration, calculated by averaging the enhancement from each wavelength.

6.4 Effects of Polarisers

This data was collected using the SLM 8000C Photon-Counting Spectrofluorometer. This outputs its data in .ols format files, or allows the data to be exported to .xls format. We used the latter and then copied the data from the .xls files into .txt files so that our programme, plotaRaph, (described earlier), could be used to plot the graphs. No significant data manipulation was required, but the programme was used because of its ability to read out the data and create the required graphs. The only part that did require some manipulation was the anisotropy graph, as the data was already in an .xls sheet. This was used to calculate the values before they were also copied into a .txt file and graphed as before.

The graph of the data we collected with all the different polariser configurations clearly demonstrates that they have an effect. Figure 6.21 is a graph of all of the polarisation data. Polarisers greatly reduced the signal, as expected, so the improvement in signal-quality had to compensate for this. On the unpolarised light dataset, there is a definite increase in the signal below ~380 nm. This is primarily caused by scattered light and clearly indicates its presence. Although that part of the spectrum could simply be ignored, varying amounts of scattered light are also present at other points, and the data continues beyond this point albeit concealed by the scattered light.

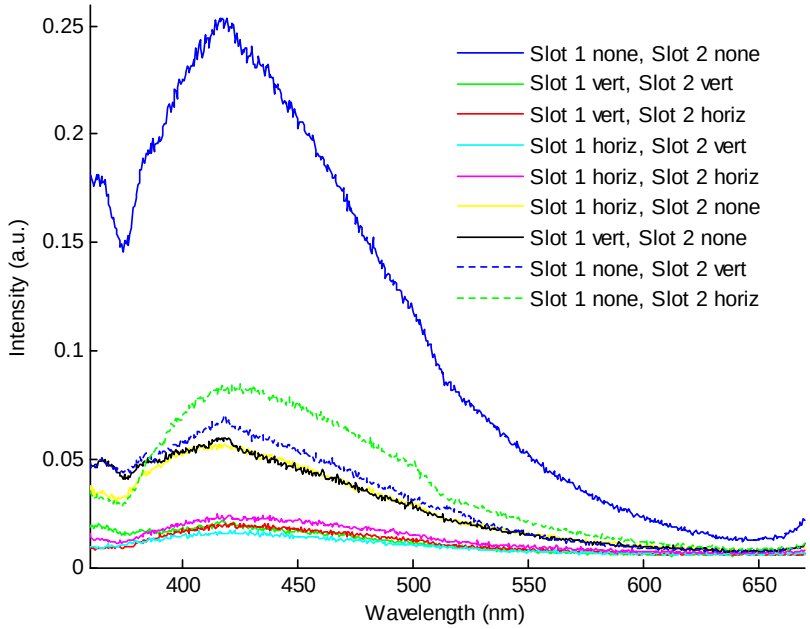

Figure 6.21: Fluorescence of whole milk powder, with double, single, and no polarisers. Excitation wavelength 350 nm.

Looking at the single- and double-polarised fluorescence in Figure 6.22, it becomes clear that this falls into two distinct groups as expected, because of the disparity in the number of polarisers. The double-polariser datasets in particular, have similar intensities; to find how the data varies we plotted each of these sets separately.

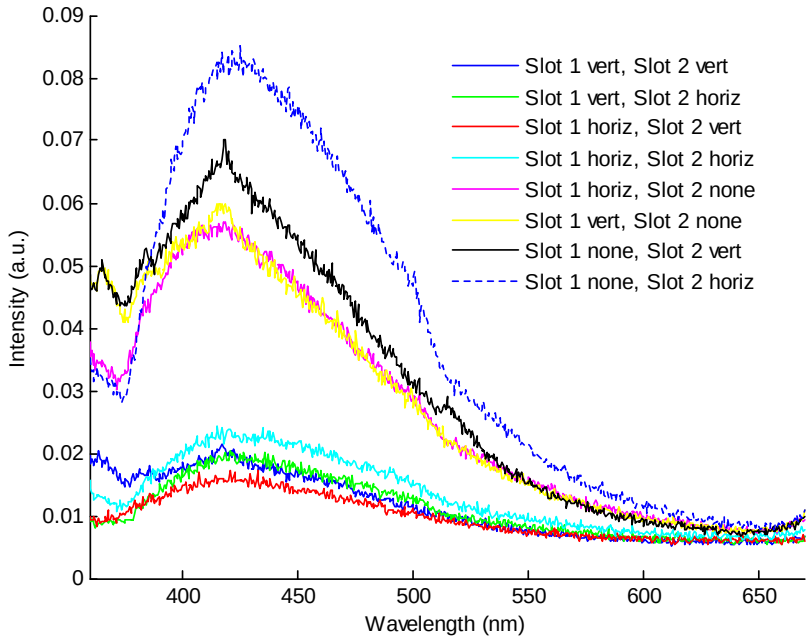

Figure 6.22: Fluorescence of whole milk powder with double, and single, polarisers, showing two distinct groups of data as expected. Excitation wavelength 350 nm.

Figure 6.23 shows the single polariser data. The red and green datasets show very similar curves, and have very prominent scattering tails. This is because they only have a vertical polariser, either in front of or behind the sample. This means the neither the scattered nor scattering light was removed, but the signal was still reduced. In fact, the green dataset shows one of the most reduced signals, yet one of the largest scattering tails. So this configuration is not desirable. The dark blue dataset has a horizontal polariser in slot 1, which reduces the amount of scattered light, as can be seen by the reduction of the scattering tail, but also reduces the signal.

The final configuration, (cyan dataset), with a horizontal polariser in the second slot, optimised the amount of light available to create the fluorescence, while removing much of the scattered light afterwards. This can be seen by the strong signal and comparatively reduced scattering tail. So this is the ideal configuration to use when only a single-polariser configuration is possible.

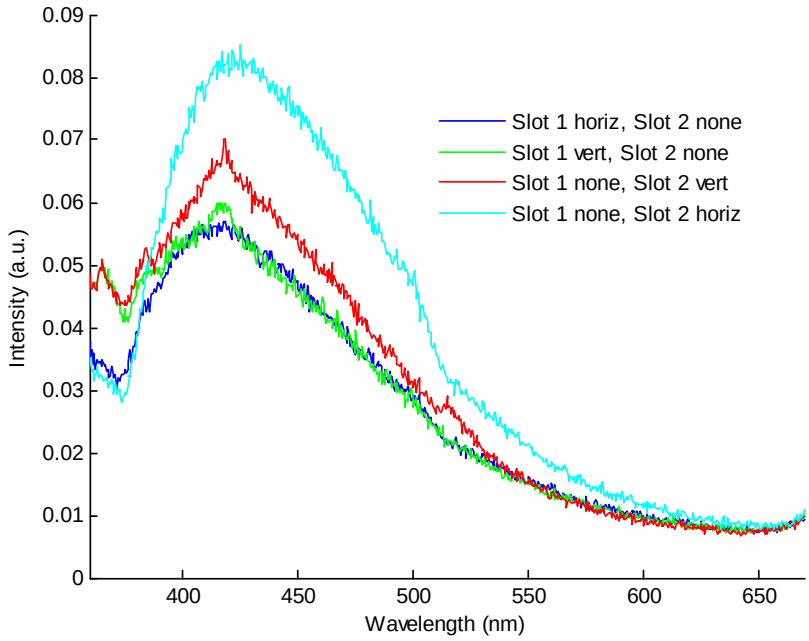

Figure 6.23: Fluorescence of whole milk powder, through different single-polariser configurations. Excitation wavelength 350 nm.

The double-polariser configurations removed even more signal than the single one, but had the potential to further optimise it. This is shown in Figure 6.24.

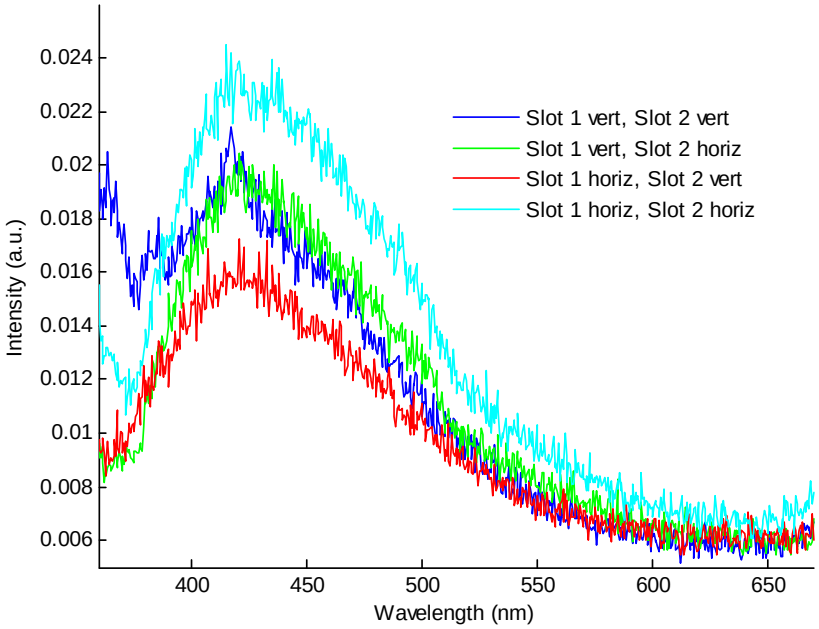

Figure 6.24: Fluorescence of whole milk powder, through different double-polariser configurations. Excitation wavelength 350 nm.

There is little variation in overall intensity, but the subtle differences in the shape of each dataset are very important. The dark blue dataset illustrates an experiment using two vertical polarisers, which remove half of the incoming light and out-going fluorescence, but fail to preferentially remove the scattered light, resulting in a scattered tail that is almost as high as the signal peak.

A vertical polariser for the incoming light and a horizontal one for the outgoing light, (green dataset), preferentially remove the scattered light. Surprisingly, it shows a higher signal peak than the red dataset, which had a horizontal incoming polariser and a vertical outgoing one, thus removing

vertical light before it reached the sample and removing the scattered light before it was even created. This difference in peak heights suggests that the polarisation of the fluorescence signal is slightly horizontally biased.

Two horizontal polarisers, (cyan dataset), preferentially removing the light that was likely to scatter before it reached the sample, and then removing any that did scatter afterwards, resulted in the strongest signal and a comparatively small scattering tail.

So the ideal double-polariser configuration for removing the scattering tail was a vertical polariser in slot 1 and a horizontal one in slot 2, because this removed most of the scatter but allowed a very strong signal to remain. The double-horizontal configuration was also very effective, resulting in a stronger signal, but also left a little more of the scattered light.

6.4.1 Anisotropy

Anisotropy is a measure of how much of the intensity measured is actually from the fluorescence. It is calculated using the equation below.

$$\text{Anisotropy} = \frac{I_{VV} - I_{VH}}{I_{VV} + I_{VH}}$$

It is essentially a measure of the scattered light with the fluorescence subtracted, the result of which is normalised. If the measured spectrum were pure fluorescence, the resulting anisotropy graph would simply show a horizontal line with a value of zero. Our anisotropy graph is shown in Figure 6.25; it shows some noise, but this was expected because all of the data in this experiment did so. This graph of anisotropy, all near zero except at the lower wavelengths where the scattered tail is, confirms that the polarisers are preferentially removing the scattering tail.

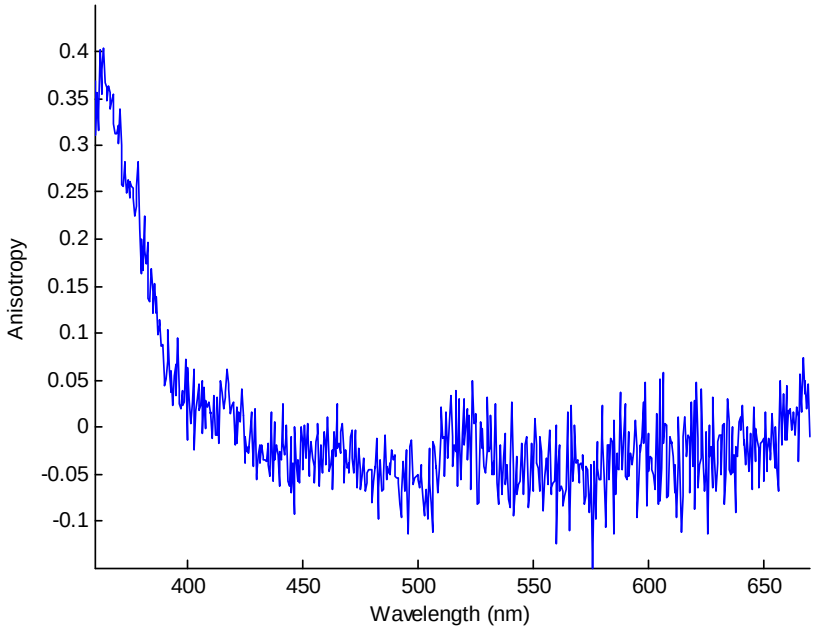

Figure 6.25: The Anisotropy, showing a near zero reading except at the scattering tail.

6.5 Linearity of Response to Enhancement Time

Data were collected using the ISS PCI Photon-Counting Spectrofluorometer, and so had to be converted to .txt files as described earlier. No datasets were collected during the enhancement process, because this was done manually to allow flexibility of the enhancement time, so the analyser programme had to be modified. We used the ExcitationProfileAnalyserV222; to make it run using only the pre- and post-enhancement data, we simply commented out the lines which read, modified or required the missing data. The collected data is summarised on Table 14.

	Peak Value ($\times 10^5$)		
Enhancement Time	**Expt. 1**	**Expt. 2**	**Mean peak value ($\times 10^5$)**
0	0.0600	0.0074	0.0337
2	0.6919	1.3541	1.0230
4	2.7120	2.6492	2.6806
6	0.7541	2.5143	1.6342
8	2.8715	2.1588	2.5152
10	3.0279	1.9434	2.4856
12	2.4918	1.3187	1.9052
14	6.4105	4.0887	5.2496

Table 14: Linearity of response to enhancement data. Peak values of enhancement from both experiments and their mean.

We added some code to the end of the analyser programme to calculate the mean of the maximum values (peak values) for each enhancement time. This

also added uncertainties and called the single plotter to plot these against the corresponding enhancement times. As this programme does not have an option to fit data and force it through zero, we took the mean values and used Excel (Microsoft, USA) to do this. We took the value calculated by Excel and put this back into the SinglePlotter to add this fitted line. This graph is shown in Figure 6.26.

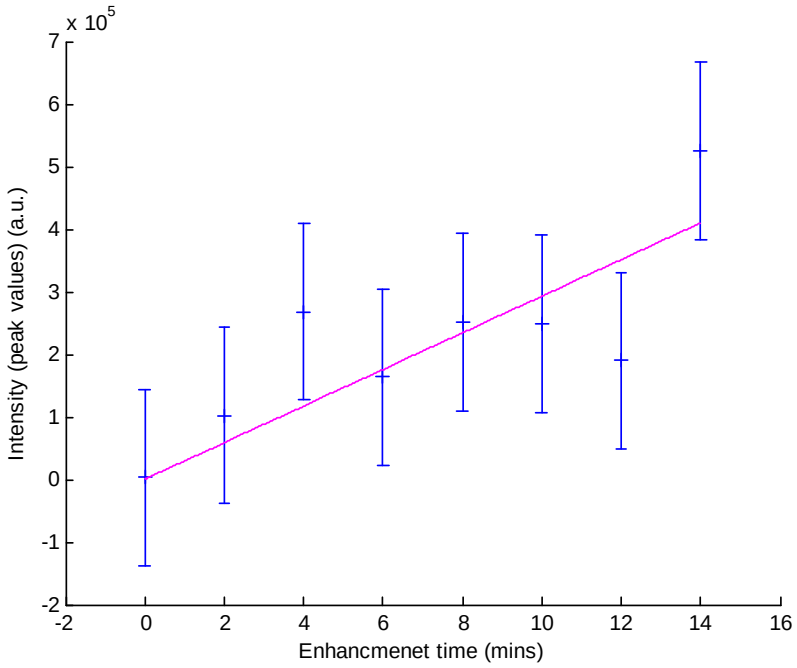

Figure 6.26: Change in enhancement, (peak values), with increasing enhancement time. Mean value of two repetitions of this experiment. Excitation wavelength 350 nm.

We found the response of CaDPA to enhancement to be $29269\ \frac{Intensity(a.u.)}{\min}$. This shows that the response is linear. Unfortunately

we were restricted to such a small range because enhancing the sample too much causes the PMTs in the fluorometer to saturate.

7 Conclusion

There are many applications for rapid bacteria detection, and yet no such detectors are currently available. The use of fluorescence for the detection of spores has been suggested by many [27], but detection technology is still in the development phase. In this thesis, some contributions towards the development of such a detector have been discussed.

We successfully demonstrated that short-wavelength UV light can be used to enhance DPA. We believe this may have applications for the identification of unknown powders, particular discrimination of spore-containing powders from those with no spores. The identification process follows the methods used in some of these experiments, where a sample's enhancement is determined by subtracting the pre- from the post-enhancement fluorescence of a sample. We successfully showed that 350 nm light was a good excitation wavelength to use, because it effectively excited the sample but caused no enhancement.

We performed a series of experiments to find the emission profile of DPA. This was achieved by creating a selection of samples, measuring their fluorescence profiles, enhancing them and re-measuring the fluorescence. This enhancement used UV light from 200 nm to 300 nm in 10 nm increments. After enhancing the sample, the fluorescence profile was re-measured using the same excitation wavelength, and the pre-enhancement profile subtracted

from it to find the enhancement. This experiment was repeated using different concentrations of DPA. The concentrations were noted to enable comparison between different measurements. We wrote a selection of MATLAB programmes to perform the required data analyses. The first was designed specifically for the analysis of single-concentration emission profiles. It was able to analyse the data in different ways, as required by the user. It created a multitude of graphs to enable the data to be interpreted, also enabling the detection of possible errors and inconsistencies, (such as corrupted samples etc.), in the measurements. A different programme was made to produce the mean emission profile from the individual ones and perform all associated analyses.

Measurements show that 210 nm light is the optimal enhancement wavelength to use. This appears to be a much more efficient, creating more enhancement per incident intensity, than other wavelengths. Our analyses suggest that using 210 nm light results in approximately 140 times more enhancement, when compared with 250 nm. Therefore, the frequently used mercury-vapour enhancement light may not be ideal for this purpose, because it utilises the 254 nm emission peak. However, if a selected wavelength from a continuous spectrum lamp is used, it may be more effective to use 270 nm, because most lamps have much greater output intensities at longer wavelengths.

Our lamp's output was approximately three orders of magnitude higher in intensity at 270 nm than at 210 nm, enhancing the fluorescence of the DPA much more quickly at 270 than at 210 nm. So, if a linearly-profiled or monochromatic enhancement light source were used, 210 nm would be ideal; otherwise it is worth considering the enhancement-lamp intensity profile when selecting the enhancement wavelength.

We measured the enhancement that could be achieved in the DPA with varying lengths of exposure to the enhancement lamp. These measurements were taken using enhancement times from 0 to 14 minutes, in two-minute increments. Measurements were restricted to less than 15 minutes because an earlier measurement had found the PMTs to saturate when enhancement of this time had taken place. The experiment was repeated to improve the quality of the results. 270 nm light was used for enhancing the sample because the stronger intensity of our lamp at this wavelength compensated for the less efficient process, (as discussed earlier). 350 nm excitation light was used to measure both the pre- and post-enhancement fluorescence of all samples. The enhancement intensity was compared to its corresponding exposure to the enhancing light, and a linear response was found.

Some light from the excitation lamp is scattered, and this, like the fluorescence signal, ends up in the PMT and causes a distortion of the measured emission

signal. This is particularly prominent at wavelengths similar to those of the excitation source, but occurs, to some extent, beyond this. Removing the scattered light is desirable to improve the signal. A series of experiments were performed using an enhanced milk powder solution to determine the extent of the effect and how best to remove it. The milk powder solution was enhanced for 30 minuets using a 150 W arc lamp. A selection of polariser configurations was examined to selectively remove the scattered light. Using a vertical/horizontal polariser configuration removed ~90% of the signal and most of the scattered light. A single horizontal polariser after the sample in the optical path only reduced the signal by two thirds, but still reduced the scattered light significantly.

We have demonstrated that the fluorescence profile of DPA can easily be enhanced, and have found that the detection protocols we have developed are effective at measuring this enhancement. Thus, we believe that this technique will be effective for distinguishing between spore-containing powders and others. We have also shown that linear relationships exist between the enhancement time or concentration and the fluorescence intensity. Once this relationship has been determined for a given detector and measurement protocol, the detector will be able to quantify the number of spores present. We also found that the quality of our results could be improved with the use of

polarisers. Our results strongly suggest that fluorescence enhancement will enable a rapid distinction to be made between suspicious powders. The research comprising this thesis has contributed towards the development of such a device.

8 References

1. Leadbetter, E.R. and J.S. Poindexter, eds. *Bacterial Activities in Perspective*. Bacteria in Nature. Vol. 1. 1985, Plenum Press: New York and London. 263.
2. Terry C. Dixon, B.S., et al., *Anthrax.* New England journal of science 1999. **341**(11): p. 815-826.
3. Doyle, M.P., L.R. Beuchat, and T.J. Montville, eds. *Food Microbiology Fundamentals and Frontiers*. 1997, ASM Press: Washington DC. 767.
4. Kunnil, J., *Identification studies of Bacillus spores using fluorescence spectroscopy* in *Department of Physics and Astronomy*. 2005, University of Canterbury: Christchurch.
5. Sharp, R.J. and A.G. Roberts, *Anthrax: the challenges for decontamination.* Journal of Chemical Technology & Biotechnology, 2006. **81**(10): p. 1612-1625.
6. Baillie, L. and T.D. Read, *Bacillus anthracis, a bug with attitude!*, . Current Opinion in Microbiology, 2001. **4**(1): p. 78-81.
7. Hugh-Jones, M.E. and V de Vos, *Anthrax and wildlife.* Revue scientifique et technique (International Office of Epizootics), 2002. **21**(2): p. 359.
8. Manchee, R.J., et al., *Bacillus anthracis on Gruinard Island.* Nature, 1981. **294**: p. 254-255.
9. Foster, D. *The Message in the Anthrax.* [cited; University of California Los Angeles (UCLA) School of Public health, Epidemiology website]. Available from: http://www.ph.ucla.edu/epi/Bioter/messageanthrax.html.
10. Guillemin, J., *Anthrax: The Investigation of a Deadly Outbreak*. 1999: University of California. 339.
11. Leonard, A.C., *Risks of publicity about bioterrorism: Anthrax hoaxes and hype.* American Journal of Infection Control, 1999. **27**(6): p. 470-473.
12. Peters, C.J. and D.M. Hartley, *Anthrax inhalation and lethal human infection.* The Lancet, 2002. **359**(9307): p. 710-711.
13. *American Anthrax outbreak of 2001 Exposure letters*. [cited; University of California Los Angeles (UCLA) School of Public health, Epidemiology website]. Available from: http://www.ph.ucla.edu/epi/Bioter/messageanthrax.html.
14. Jernigan, D.B., et al., *Investigation of Bioterrorism-Related Anthrax, United States, 2001: Epidemiologic Findings.* Emerging Infectious Disease, 2002. **8**(10): p. 1669-1673.

15. Canter, D.A., *Addressing Residual Risk Issues at Anthrax Cleanups: How Clean is Safe?* Journal of Toxicology and Environmental Health, Part A, 2005. **68**(11-12): p. 1017-1032.
16. Min, J.W., J.Y. Lee, and R.A. Deininger, *Simple and rapid method for detection of bacterial spores in powder useful for first responders* Journal of Environmental Health, 2006. **64**(8): p. 34-37.
17. Farquharson, S. and W. Smith, *Differentiating bacterial spores from hoax materials by Raman Spectroscopy*, Real-Time Analyzers, 87 Church Street, East Hartford, CT 06108.
18. Series, U.S.F.A.T.R., *Fire Department Response to Biological Threat at B'nai B'rith Headquarters Washington, DC, April 1997*, United States Fire Administration.
19. Kristof, N.D. *The Anthrax Files*. [cited; University of California Los Angeles (UCLA) School of Public health, Epidemiology website]. Available from: http://www.ph.ucla.edu/epi/bioter/theanthraxfiles.html.
20. Ryu, C., et al., *Sensitive and Rapid Quantitative Detection of Anthrax Spores Isolated From Soil Samples By Real-Time PCR.* Microbiology and Immunology, 2003. **47**(10): p. 693-699.
21. Kimberly Quinlan Lindsey, P.D. and M.S.P.H. Stephen A. Morse, Ph.D., *Basic Laboratory Protocols for the Presumptive Identification of Bacillus anthracis*. 2001.
22. Church, B.D. and H. Halvorson, *Dependence of the heat resistance of the bacterial endospores on their dipicolinic acid content.* Nature, 1959. **183**: p. 124-125.
23. Stieman, T.A. and W.L. Nichlson, *Role of Dipicolinic Acid in the Survival of Bacillius subtilis Spores Exposed to Artifical and Dolar UV Radiation.* Applied and Environmental Microbiology, 2001. **67**(3): p. q274-1279.
24. Amann, R.I., W. Ludwig, and K.H. Schliefer, *Phylogenitic Identification and in situ detection of individual microbial cells without cultivation.* Microbiology Review, 1995. **59**(1): p. 143-169.
25. Hugenholtz, P., B.M. Geobel, and N.R. Pace, *Impact of culture-independent studies on the emerging phylogenetic view of bacterial diversity.* journal of Bacteriology, 1998. **180**(18): p. 4765-4774.
26. *Manufaturer web site (ISS)*. [cited; Available from: http://www.iss.com/products/pc1/diagram.html.
27. Reinisch, L. and B.V. Bronk, *Variability of Steady State Bacterial Fluorescence with respect to Growth Conditions.* Applied Chemistry, 1993. **47**(4): p. 436-440.

9 APPENDIX 1: Data Plotter

This plotter was written to analyse and plot data being collected. Being the first of the series of programmes we wrote, it is the most generic and was used for a wide variety experiments. It is versatile, and can be used for plotting such things as the warm-up times, as well as emission profiles. For the latter, more specific programmes were usually used – see Appendices 2 and 3.

This programme has the option for all variables to be hard-coded. Usually a combination is used, i.e. some variables, which do not change for the specific data being worked with, are hard-coded and those that are frequently changed are varied by using the user input option.

plotaRaph.m: the main running file. This collects the data and calls the other required functions, and then plots the desired graphs.

```
% This run file will plot a selection of data sets onto one
graph
% to use it, input a file name containing the names of the
.txt files that
% should be plotted it also calls the plot decorator.m file
which allows
% axes legends and other standard labels to be placed on the
graph. it also
% has optional datamanipulator, maxvaluefinder and
datasmoother
% functions attached which can be used if desired.

% Author: Raphael Nolden © 2007
% plotaRaph Version 1.0 developed from the multi-plotter I
wrote for my
% fluorometer data
% Version number 1
% Date Created: 23 Jan 2006
% CHANGES from fluorometer plotter
% takes out parts which are very specific to my data.
```

```
% generalises it to work with different types of data files
incl different
% header lengths and different number of cols in data
% also so it works with datasets of different lengths all on
one graph

% INITIALISE VARIABLES

% clear variables and initialise global variables
clear all
clc
global colourcounter linecounter

% initialise counters
colourcounter = 0;
linecounter = 1;
datasetcounter = 0;

% % To hard-code variables which are always the same, un-
comment them here and comment out there
% % respective input later on
% namefile = 'excitation profile.txt';
% maxcheck = 1;
% datamanipulatorcheck = 1;
% datasmoothercheck = 1;
% noofcolumns = 5;
% headerlength = 22;
% xcol = 1;
% ycol = 3;
% usefulrange = [1 171];
% smoothingsize = 7;
% columntosmooth = 2;
% plotmaxvaluecheck = 1;
% maxarray = [250 255 260 265 270 275 280 285 290 295 300];

% COLLECT REQUIRED INFORMATION FROM USER

% input data name file
% file should contain one file name per line with no quotes or
anything else around it.
% also ensure there are no spaces or empty lines at the end
namefile = input('Input file name of .txt file containing
names of data files to be plotted: ','s');

% check which optional programmes should be run
(maxvaluefinder,
% maxvalueplotter, datamanipulator, datasmoother)
maxcheck = input('To collect the peak value from each dataset
press 1 then enter, otherwise press enter: ');
```

```
datamanipulatorcheck = input('To use the datamanipulator press
1 then enter, otherwise press enter: ');
datasmoothercheck = input('To use the datasmoother press 1
then enter, otherwise press then enter: ');

% get info about the data required to extract and plot it
noofcolumns = input('Input the number of columns of data in
the data files: ');
headerlength = input('input the number of line taken up by the
header: ');
xcol = input('Input the column number to be used as the x
axis: ');
ycol = input('Input the column number to be used as the y
axis: ');

% collect useful range is maxchacker is being run
if maxcheck == 1
    usefulrange = input('Input useful range for finding
maxvalue as a row vector [minrow maxrow] press 0 to use all
datapoints: ');
end

% if datasmoother is to be run, collect required vbariables
if datasmoothercheck == 1
    smoothingsize = input('Input odd number of point greater
than 2 that should be used for smoothing: ');
    columntosmooth = input('Input number of the column to be
smoothed: ');
end

% Check if max value should be collected, and, if so, whether
these should be
% plotted and collect an x-array to plot against
if maxcheck == 1
    plotmaxvaluecheck = input('To plot the maxvalues press 1
then enter otherwise press enter: ');
    if plotmaxvaluecheck == 1
        maxarray = input('Input x array for max values [ ]:
');
    end
end

% Initialise figure and set up for multiple plots
figure(1)
hold on


% PERFORM DATA EXTRACTION MANIPULATION AND PLOTTING
```

```
% make file so it is ready for reading by MATLAB
fid = fopen(namefile);

% Extract filenames from the file until all have been read and
their data
% has been extracted
while 1
    % select next file to extract data from
    filename = fgetl(fid);

    % check if end of file containing names of datafiles has
been reached
    if filename == -1
        break
    end

    % count number data sets plotted to enable custom legend
to be made
    datasetcounter = datasetcounter + 1;

    % collect file names of datafiles for default legend
    legendnamescell{datasetcounter} = filename;

    % run data extraction function. Extracts data from a text
file ignoring
    % header rows
    [data,datalength] =
DataExtractor(filename,headerlength,noofcolumns);

    % check if the datasmoother should be run. this smoothes
data by averaging
    % over smoothingsize number of elements
    if datasmoothercheck == 1
        [data] =
DataSmoother(data,datalength,smoothingsize,columntosmooth);
    end

    % check if optional data manipulator should be run. this
can be
    % used if the data needs any form of manipulation or
processing after extraction
    if datamanipulatorcheck == 1

        % run the datamanipulator
        [data] = DataManipulator(data,ycol,datalength);
    end
```

```
    % check if max values are to be collected and runmaxvalue
finder
    if maxcheck == 1

        % Run the max value finder
        [maxvalue] =
MaxFinder(data,ycol,usefulrange,datalength);
        % save maxvalue into array
        maxvaluearray(datasetcounter) = maxvalue;
    end

    %plot data using data plotter
    Plotter(data,xcol,ycol);
end

% DECORATE PLOT AND PLOT MAX VALUES IF REQUIRED

% finish multiple plots and label the graph using the
plotdecorator
hold off
PlotDecorator(legendnamescell,datasetcounter);

% check if maxvalue plot should be plotted
if plotmaxvaluecheck == 1
    MaxValuePlotter(maxvaluearray,maxarray)
end
```

DataExtractor.m: extracts the data and stores it into an array.

```
function [data,datalength] =
DataExtractor(filename,headerlength,noofcolumns);
% extracts the data from the text file for a given a filename
% filename should be entered as a string e.g. 'filename.txt'
% it outputs a data array called data which contains the data
in columns
%
% calling sequence: [data,datalength] =
DataExtractor(filename,headerlength,noofcolumns)
%
% Input variable:
% filename        string containing the full name including
extention of the file containing the data (.txt)
% headerlength    number of rows in taken up by the header
% noofcolumns     number of columns of data
%
% Output Variables
```

```
% data             data array containing all columns and rows
of data
% datalength       number of rows in the data array
%
% Author: Raphael Nolden © 2007

% initialise variables
endchecker = 0;
jj = 0;

% create reference to file that Data is being extracted from
fid = fopen(filename);

% loop through header lines to be deleted
for ii = 1:1:headerlength
    delete = fgetl(fid);
end

% make loop to extract useful data
while endchecker == 0;
    jj = jj + 1;
    % use a try catch so that it will just automatically stop
when it
    % reaches the end of the data and not crash
    try
        % extract line and convert it to a number to put into
array
        thisline = fgetl(fid);
        numline = str2num(thisline);

        % put useful data into an array
        for kk = 1:1:noofcolumns
            % put useful data into an array
            data(jj,kk) = numline(kk);
        end
    catch
        % shows that the end has been reached and so stops the
while loop
        endchecker = 1;
    end
end

% calculates the datalength
datalength = length(data);
```

DataSmoother.m: smoothes the data, by taking the average of a desired number of data-points and saves it to the central point.

```
function [data] =
DataSmoother(data,datalength,smoothingsize,columntosmooth);
% this function smoothes data sets to reduce noise. It
calculates the average
% of a desired number of terms (smoothingsize) and saves the
resultant
% number into a new array. It uses an odd number of terms to
smooth and
% puts the number into the middle place. beginning and end
numbers use the
% max of eachside on any side of the centre value.
%
% calling sequence: [data] =
DataSmoother(data,datalength,smoothingsize,columntosmooth)
%
% Input variables
% Data              an array of numbers, one column of which
is to be smoothed
% datalength        number of rows in the array
% smoothingsize     number of terms to be averaged
% columntosmooth    column in the array that is to be
smoothed.
%
% Output variables
% data              data with the desired column replaced with
the smoothed one
%
% Author: Raphael Nolden © 2007

% check that an appropriate number has been entered for
smoothingsize
if smoothingsize <= 1 || rem(smoothingsize,1)~=0
    error('smoothing size must be a positive integer and
greater than 2')
end

% check smoothingsize to ensure it is an odd number otherwise
assigns a new value of specified + 1
if rem(smoothingsize,2) == 0
    smoothingsize = smoothingsize + 1;
    fprintf('the value entered for smoothingsize is not an odd
number so %2u was used instead\n',smoothingsize)
end

% extracts the desired column of data out of the data array
smoothingdata = data(:,columntosmooth);
```

```
% define how many values are on each side of the centre point
eachside = (smoothingsize - 1)/2;
backwardcounter = eachside;

% perform first 'eachside' number of iterations
for ii=1:1:eachside
    thissample = smoothingdata(1:ii+eachside,1);
    smoothdata(ii,1) = sum(thissample)/(ii+eachside);
end

% performs smoothing where smoothingsize number of variables
are available
% to do smoothing over
for jj = eachside+1:1:(datalength-eachside)
    % create subarray of smoothingsize points to average
    thissample = smoothingdata(jj-eachside:jj+eachside,1);
    % average data and save data into array
    smoothdata(jj,1) = (sum(thissample))/smoothingsize;
end

% perform smoothing on final data points
for kk=datalength-eachside+1:1:datalength
    thissample = smoothingdata(kk-eachside:datalength,1);
    smoothdata(kk,1) =
sum(thissample)/(backwardcounter+eachside);
    backwardcounter = backwardcounter - 1;
end

% put smoothed data back into data array
data(:,columntosmooth) = smoothdata;
```

DataManipulator.m: can be used to perform some desired data manipulation if required.

```
function [data] = DataManipulator(data,ycol,datalength);
% this programme can be used to manipulate data. programme in
whatever is required.
% manipulation you want to do.
%
% Input:
% data          data array being used
% ycol          column you wish to manipulate
% datalength    number of rows in data array
%
% Output
```

```
% data          data array with the manipulated data saved
into ycol
%
% Author: Raphael Nolden © 2007
% extract column to be manipulated
mandata = data(:,ycol);

% perform calculation
intermeadiatevariable = mandata *1;

% put back into data array
data(:,ycol) = intermeadiatevariable;
```

MaxFinder.m: Finds the maximum value in the dataset; this can be restricted to a certain domain of the data.

```
function [maxvalue] =
MaxFinder(data,ycol,usefulrange,datalength);
% Finds the max value of the column (ycol) of a data array
(data)
%
% calling sequence: [maxvalue] = MaxFinder(data,ycol)
%
% Input:
% data          data array
% ycol          column of which the maxvalue is to be found
% usefulrange   1X2 array containing min and max datapoints of
interest.
% useful(1,1)   is first useful point
% useful(1,2)   is last useful point
%
% Output:
% maxvalue    maxvalue of this column of the data array
%
% Author: Raphael Nolden © 2007

% check what type of data range is required and that an
appropriate value
% has been input
if usefulrange == 0
    % define what is considered to be the useful range of data
to find the
    % max in. this is used by default unless another range is
specified
    usefulrange = 450-390;

    % calculate useful data range to extract 390-450nm
```

```
    beforeuseful = 391 - data(1,1);
    endofuseful = beforeuseful + usefulrange;
    % cut data down to data range of interest 390-450nm
    usefuldata = data(beforeuseful:endofuseful,ycol);

    usefuldata = data(:,ycol);
elseif usefulrange(1,1) < 1 || usefulrange(1,2) > datalength
    usefuldata = data(:,ycol);
    disp('Inappropriate useful data range defined. All data
points will be used instead')
else
    % define what range is to be used
    usedrange = usefulrange(1,1)-usefulrange(1,2);

    % extract useful part of data
    usefuldata = data(usefulrange(1,1):usefulrange(1,2),ycol);
end

% finds max value of previously defined useful data range
maxvalue = max(usefuldata(:,1));
```

Plotter.m: plots the data.

```
function [] = Plotter(data,xcol,ycol);
global colourcounter linecounter
%
% Plotter
% plots the data on the columns of the data array as specified
by the
% variables xcol and ycol
% takes the data matrix, extracts the columns required for the
plot and
% plots them. uses a different colour and linetype combination
for each
% subsequent plot.
%
% requires global variables colourcounter and linecounter to
initialise the
% variables only the first time the programme is called
%
% calling sequence: Plotter(data,xcol,ycol)
%
% Input variables:
% data      data array
% xcol      column number to be used as the x-axis
% ycol      column number to be used as the y-axis
%
```

```
% NO Output variable
%
% Author: Raphael Nolden © 2007

% create a cell array of colours for the plots
colourange = {'blue' 'green' 'red' 'cyan' 'magenta' 'yellow'
'black'};
linetypes = {'-' ':' '--' '-.'};

% extract column vectors to be plotted
xarray = data(:,xcol);
yarray = data(:,ycol);

% increment colourcounter and linetype counter as appropriate
if colourcounter >= length(colourange)
    colourcounter = 1;
    if linecounter == length(linetypes)
        % reset linecounter if the end of the line array has
been reached
        linecounter = 1;
    else
        % increment to linecounter
        linecounter = linecounter + 1;
    end
else
    % change colour for next data set
    colourcounter = colourcounter+1;
end

% select the next colour and linestyle and create single
string
colourcell = colourange(colourcounter);
linecell = linetypes(linecounter);
colourstr = char(colourcell);
linestyle = char(linecell);
thistyle = strcat(colourstr,linestyle);

% plot graph using x-array and y-array and concatenated
linestyle
plot(xarray,yarray,thistyle)
```

PlotDecorator.m: adds axes and a title to the graph, and also creates a legend if desired.

```
function [] = PlotDecorator(legendnamescell,datasetcounter);
%
```

```
% plot decorator
% should be used after plotting graphs. it takes input
% from the user for things such as title and axis labels and
then allows
% custom axis and legends to be defined otherwise it uses the
defaults
% including the default legend entered in as the
legendnamescell
%
% calling sequence:
PlotDecorator(legendnamescell,datasetcounter)
%
% Input variables:
% legendnamescell     cell array of legend names
% datasetcounter      number of datasets plotted
%
% NO Output variables

% global legendnamescell legendcounter asks for inputs and
labels the
% graph. also sets other parameters.

% % initialise checker variables
% alldefaultchecker = 5;
% customaxischeck = 5;
% customlegendcheck = 5;

% input and set standard graph labels
graphtitle = input('Input title for graph: ','s');
title(graphtitle)
xaxislabel = input('Input x-axis label: ','s');
xlabel(xaxislabel)
yaxislabel = input('Input y-axis label: ','s');
ylabel(yaxislabel)

% allows user to choose either default or custom axis and
legend for graph
alldefaultcheck = input('To use default axis and legend names
press enter, otherwise press 1 then enter: ');

if alldefaultcheck == 1
    % determines which parameters are to be custom
    customaxischeck = input('To enter custom axis values press
1 then enter, otherwise press enter: ');
    customlegendcheck = input('To input custom legend names
press 1 then enter, otherwise press enter: ');

    % determine type of axis desired and set
    if customaxischeck == 1
```

```
        axstr = input('Please enter the desired axis values as
a vector [xmin xmax ymin ymax]: ');
        axis([axstr])
    end

    % determine type of legend required and create it
    if customlegendcheck == 1

        % inform user how to input custom legend names
        disp('Enter one legend title at a time, in the order
they appeared in the data name file')

        % collect as many legend names as there are datasets
        for ii = 1:1:datasetcounter
            thislegend = input('Input legend name: ','s');
            legendnamescell{ii} = thislegend;
        end
    end
end

% put legend onto graph using legend chosen or defined earlier
legend(legendnamescell,'location','north')
legend boxoff
```

MaxValuePlotter.m: plots the max values collected by the MaxFinder.m.

```
function [] = MaxValuePlotter(maxvaluearray,maxarray);
% function to plot the max value array. takes in maxvalue
array and plots
% it against y-array
%
% Input:
% maxvaluearray     array of max values to be plotted
% maxyarray         array of y values
%
% Author: Raphael Nolden © 2007

% define new figure and start hold on
figure(2)

% plot data
plot(maxarray,maxvaluearray,'+',maxarray,maxvaluearray,'-')

% input and set standard graph labels
graphtitle = input('Input title for max value graph: ','s');
title(graphtitle)
```

```
xaxislabel = input('Input x-axis label: ','s');
xlabel(xaxislabel)
yaxislabel = input('Input y-axis label: ','s');
ylabel(yaxislabel)
```

10 APPENDIX 2: Single-Concentration Emission Profile Analysis Programme

This programme was written to analyse the data being collected, to find the emission profile of the DPA. It takes in all the required values for the variables and extracts, analyses and plots the data.

This programme has the option for all variables to be hard-coded. Usually a combination is used, i.e. some variables which do not change for the specific data being worked with are hard-coded and those that are being changed are varied by using the user input option.

This programme uses some of the same function we wrote for the other programmes. They are only shown once, at their first occurrence, to prevent repetition.

EmissionProfileAnalyserV222.m: is the main running programme. It collects the initial values for variables and calls the other functions when required.

```
% run file for analysing and comparing multiple data sets.
specifically, this
% is to analyse the enhancement profile of my fluorometer data
%
% all input and data files should be .txt files. the 4 files
containing the
% names of the data files should only contain the names with
no empty rows
% or spaces at the end.
%
```

```
% content of arrays:
%
% *data:
% col 1     wavelength (nm)
% col 2     excitation (counts)
% col 3     emission (counts)
% col 4     std. dev. excitation
% col 5     std. dev. emission
% col 6     smoothed data (smoothed col 2)
% col 7     manipulated data (col 3, normalised for lamp
intensity)
%
% enhancement
% col 1     wavelength (nm)
% col 2     enhancement
% col 3     normalised enhancement
%
% Author: Raphael Nolden © 2007
%
% Revision
% Version 2.1
% updates include saving integration of excitation into an
array, plotting these
% values, and integrating the total enhancement and plotting
this with
% respect to each wavelength as an alternative way of looking
at max value.
%
% Version 2.1.1
% added option to plot either enhancement, pre- or post-data,
using an if
% statement controlled by an input variable or hard-coded
%
% Version 2.2
% upgraded plotdecorator and maxvalueplotter so they take in
title and axis names
% to allow this to be automated. Automate the titles and axes
of plots
% where they are always the same. enhance the features in
choosing pre-enhancement
% post-enhancement or enhancement, and automate axes.
% made it possible to plot pre- and post-enhancement adjusted
for variations
% in lamp intensity
% also offers option to use background light to calibrate.
% also fixed double-calibration for the change in enhancement-
lamp
% intensity.
% checked all logic in programme!
%
```

```
% Version 2.2.1
% Changed initial input asking for type of EXPT so it is
readable
%
% Version 2.2.2
% Updated graph title creation so that it specifies if it has
been
% standardised for lamp output variation

% INITIALISE VARIABLES

% clear variables and initialise global variables
clear all
clc
global colourcounter linecounter

% initialise counters
colourcounter = 0;
linecounter = 1;
datasetcounter = 0;
calibratewithexcitationcheck = 0;

% INITIALISE VARIABLES BY HARD-CODING THEM IN
% To hard-code in variables which are always the same, un-
comment them here and comment out the
% respective input later on
% maxcheck = 1;
% datamanipulatorcheck = 1;
% calibratewithexcitationcheck = 1;
datasmoothercheck = 1;
mainplottype = 1;
noofcolumns = 5;
headerlength = 23;
wavelengthcol = 1;
enhancementcol = 3;
usefulrange = 0;
smoothingsize = 9;
columntosmooth = 2;
smoothedcolumn = 6;
plotmaxvalueintegratedcheck = 1;
plotmaxvaluepeakcheck = 1;
plotintegratedlampintesitycheck = 1;
preandpostcheck = 0;
maxarray = [200 210 220 230 240 250 260 270 280 290 300];
manipsavetocol = 7;
emissioncol = 3;
legendnames = 'legendnames.txt';

% % % COLLECT REQUIRED INFORMATION FROM USER
```

```
% %
% % % input data namefile
% % % file should contain one file name per line with no
quotation marks or anything else around it.
% % % also ensure there are no spaces or empty lines at the
end
% % namefile = input('input file name of .txt file containing
names of data files to be plotted: ','s');
% %
% % % check which optional programmes should be run
(maxvaluefinder,
% % % maxvalueplotter, datamanipulator, datasmoother)
% % maxcheck = input('To collect the peak value from each
dataset press 1 then enter, otherwise press enter: ');
% % datamanipulatorcheck = input('to use the datamanipulator
press 1 then enter, otherwise press enter: ');
% % datasmoothercheck = input('to use the datasmoother press 1
then enter, otherwise press enter: ');
% % preandpostcheck = input('to plot the pre- and post-
enhancement of each wavelength, press 1 then enter, otherwise
press enter: ');
% %
% % % get info about the data required to extract and plot it
% % noofcolumns = input('input the number of columns of data
in the data files: ');
% % headerlength = input('input the number of lines taken up
by the header: ');
% % % xcol = input('input the column number to be used as the
x axis: ');
% % % ycol = input('input the column number to be used as the
y axis: ');
% % enhancementcol = input('input the column number of the
enhancement to be plotted: ');
% % emissioncol = input('input the column containing the
emission data: ');
% % wavelengthcol = input('input the number of the column
containing the wavelengths: ');
% % legendnames = input('input name of .txt file containing
the legend names incl. the extension: ','s');

% Display the options for the plot type and get user input for
which one is
% required
disp('To plot pre-enhancement values press 1')
disp('To plot post-enhancement values press 2')
disp('To plot the enhancement press 3')
disp('To plot pre-enhancement values adjusted for variation in
lamp intensity press 4')
disp('To plot post-enhancement data adjusted for variations in
lamp intensity press 5')
```

```
disp('To plot enhancement adjusted for variations in lamp
intensity press 6')

mainplottype = input('Please enter the number corresponding to
the plot type above then press enter: ');

% if one of the calibrated plots has been chosen set
calibration check to 1
if mainplottype > 3
    calibratewithexcitationcheck = 1;
end

% % % collect useful range if maxchecker is being run
% % if datamanipulatorcheck == 1
% %     manipsavetocol = input('Input column to which
manipulated data should be saved: ');
% % end
% %
% % % if datasmoother is to be run, collect required variables
% % if datasmoothercheck == 1
% %     smoothingsize = input('input odd number of points
greater than 2 which should be used for smoothing: ');
% %     columntosmooth = input('Input number of the column to
be smoothed: ');
% %     smoothedcolumn = input('Input column where smoothed
data should be saved: ');
% % end
% %
% % % Check if max values should be collected and if so,
whether these should be
% % % plotted and collect an x-array to plot against
% % if maxcheck == 1
% %     usefulrange = input('Input useful range for finding
maxvalue as a row vector [minrow maxrow] press 0 to use all
datapoints: ');
% %     plotmaxvaluecheckpeak = input('To plot the maxvalues
(peak) press 1 then enter otherwise press enter: ');
% %     plotmaxvaluecheckintegrated = input('To plot the
maxvalues (int) press 1 then enter otherwise press enter: ');
% %     maxcol = input('Column from which max value should be
extracted: ');
% %     if plotmaxvaluecheck == 1
% %         maxarray = input('Input x array for max values [
]: ');
% %     end
% % end

% Initialise figure and set up for multiple plots
figure()
```

```
hold on

% read in file names of all three datasets. each file should
contain a list
% of file names for the corresponding dataset
prenamefile = 'preenhancement.txt';
during1namefile = 'during1enhancement.txt';
during2namefile = 'during2enhancement.txt';
postnamefile = 'postenhancement.txt';

% define files which contain the names of the data files for
each type
fidpre = fopen(prenamefile);
fidduring1 = fopen(during1namefile);
fidduring2 = fopen(during2namefile);
fidpost = fopen(postnamefile);

% PERFORM DATA EXTRACTION, MANIPULATION AND PLOTTING

% Extract filenames from the file until all have been read and
their data
% has been extracted
while 1
    % select the next filename of each type to extract data
from
    thisprename = fgetl(fidpre);
    thisduring1name = fgetl(fidduring1);
    thisduring2name = fgetl(fidduring2);
    thispostname = fgetl(fidpost);

    % check if end of file containing names of data files has
been reached
    if thisprename == -1
        break
    elseif thisduring1name == -1
        break
    elseif thisduring2name == -1
        break
    elseif thispostname == -1
        break
    end

    % count number datasets plotted to enable custom legend to
be made
    datasetcounter = datasetcounter + 1;

    % run data extraction function for each of the four data
files and save
    % the data into four different arrays
```

```
    [predata,predatalength] =
DataExtractor(thisprename,headerlength,noofcolumns);
    [during1data,duringdatalength] =
DataExtractor(thisduring1name,headerlength,noofcolumns);
    [during2data,duringdatalength] =
DataExtractor(thisduring2name,headerlength,noofcolumns);
    [postdata,postdatalength] =
DataExtractor(thispostname,headerlength,noofcolumns);

    % check if calibration for excitation should be done. this
is for the
    % plotting pre and post data
    if calibratewithexcitationcheck == 1
        % run the datamanipulation programme which divides the
emission data by
        % smoothed excitation data
        [predata] = DataManipulator(predata,manipsavetocol);
        [postdata] = DataManipulator(postdata,manipsavetocol);
    end

    % check if datasmoother should be run
    if datasmoothercheck == 1
        % run the datasmoother on the pre and post file
storing the smoothed data
        % into the smoothed data column
        [predata] =
DataSmoother(predata,predatalength,smoothingsize,columntosmoot
h,smoothedcolumn);
        [postdata] =
DataSmoother(postdata,postdatalength,smoothingsize,columntosmo
oth,smoothedcolumn);
    end

    % create a new array to save the enhancement values into
    % save in wavelength
    enhancement(:,1) = predata(:,1);

    % calculate the enhancement and save into enhancement
array
    % (enhancement = post - pre both adjusted for intensity of
lamp
    enhancement(:,2) = postdata(:,emissioncol) -
predata(:,emissioncol);

    % integrate the enhancement light and saves to an array
    intenhancement(datasetcounter) = (sum(during1data(:,2)) +
sum(during2data(:,2)));

    % normalise for the excitation channel if required
```

```
    if calibratewithexcitationcheck == 1
        % calculate the normalised enhancement
        enhancement(:,3) =
enhancement(:,2)./intenhancement(datasetcounter);
    else
        enhancement(:,3) = enhancement(:,2);
    end

    % integrates the enhancement and saves to an array
    intnormalen(datasetcounter) = sum(enhancement(:,3));

    % Run the max value finder
    [maxvalue] =
MaxFinder(enhancement,3,usefulrange,predatalength);

    % save maxvalue into array
    maxvaluearray(datasetcounter) = maxvalue;

    % save all data into a cell array
    alldata{datasetcounter,1} = predata;
    alldata{datasetcounter,2} = during1data;
    alldata{datasetcounter,3} = during2data;
    alldata{datasetcounter,4} = postdata;
    alldata{datasetcounter,5} = enhancement;

    % plot desired data type
    if mainplottype == 1
        % plot pre-enhancement values
        Plotter(predata,wavelengthcol,emissioncol);

    elseif mainplottype == 2
        % plot post-enhancement values
        Plotter(postdata,wavelengthcol,emissioncol);

    elseif mainplottype == 3
        % plot enhancement (wavelength vs enhancement)
        Plotter(enhancement,wavelengthcol,enhancementcol);

    elseif mainplottype == 4
        % check if calibration was done; if not, the
calibrated values cannot be plotted
        if calibratewithexcitationcheck == 1
            % plot pre-enhancement values (normalised for lamp
intensity)
            Plotter(predata,1,manipsavetocol);
        else
            % display error because calibrated graph selected
but
            % calibration not run
```

```
            error('cannot display graph of calibrated values
because using calibration was not selected')
        end

    elseif mainplottype == 5
        if calibratewithexcitationcheck == 1
            % plot post enhancement values (normalised for
lamp intensity)
            Plotter(postdata,wavelengthcol,manipsavetocol);
        else
            % display error because calibrated graph selected
but
            % calibration not run
            error('cannot display graph of calibrated values
because using calibration was not selected')
        end

    elseif mainplottype == 6
        if calibratewithexcitationcheck == 1
            % plot data using data plotter (wavelength vs
normalised enhancement)
            Plotter(enhancement,wavelengthcol,enhancementcol);
        else
            % display error because calibrated graph selected
but
            % calibration not run
            error('cannot display graph of calibrated values
because using calibration was not selected')
        end

    else
        error('Inappropriate plot type selected, please select
a number from 1-6')
    end
end

% DECORATE GRAPH AND MAKE OTHER GRAPHS AS REQUIRED

% CREATE DEFAULT LEGEND ARRAY
% identify file containing legend names
fidlegend = fopen(legendnames);

% extract legend names
for kk = 1:1:datasetcounter

    % extract the next legend name
    thisline = fgetl(fidlegend);
```

```
    % if there are not enough names in the legend name .txt
file, it won't crash
    % but will state this on the graph
    if thisline == [-1]
        thisline = 'Not enough legend names in
legendnames.txt';
    end

    % put name into an array
    legendnamescell{1,kk} = thisline;
end

% CREATE LABELS FOR MAIN PLOT
% create appropriate plot labels for main plot depending of
type of plot
% being made
if mainplottype == 1
    graphtitle = 'Pre Enhancement';
    xaxislabel = 'Wavelength (nm)';
    yaxislabel = 'Counts';
elseif mainplottype == 2
    graphtitle = 'Post Enhancement';
    xaxislabel = 'Wavelength (nm)';
    yaxislabel = 'Counts';
elseif mainplottype == 3
    graphtitle = 'Enhancement';
    xaxislabel = 'Wavelength (nm)';
    yaxislabel = 'Counts';
elseif mainplottype == 4
    graphtitle = 'Pre Enhancement normalised for lamp
intensity variations';
    xaxislabel = 'Wavelength (nm)';
    yaxislabel = 'Counts';
elseif mainplottype == 5
    graphtitle = 'Post Enhancement normalised for lamp
intensity variations';
    xaxislabel = 'Wavelength (nm)';
    yaxislabel = 'Counts';
elseif mainplottype == 6
    graphtitle = 'Enhancement normalised for lamp intensity
variations';
    xaxislabel = 'Wavelength (nm)';
    yaxislabel = 'Counts';
else
    error('mainplottype not valid, (must be between 1-6)')
end

% finish multiple plots and label the graph using the
plotdecorator
hold off
```

```
PlotDecoratorV11(legendnamescell,datasetcounter,0,graphtitle,x
axislabel,yaxislabel);

% PLOT PRE- AND POST-ENHANCEMENT DATA
% check if pre and post enhancement data should be plotted
if preandpostcheck == 1
    % plot the pre- and post-enhancement emission, one graph
for each wavelength
    for jj = 1:1:datasetcounter
        figure

plot(enhancement(:,1),alldata{jj,1}(:,3),'magenta',enhancement
(:,1),alldata{jj,4}(:,3),'black:')
        title(legendnamescell{1,jj})
        xlabel('Wavelength (nm)')
        ylabel('Counts')
        % % uncomment this and define axis size if all graphs
should have same axes
        % axis([380 550 0 30e4])
        text(450,3.0e4,'pre = pink, post = dotted black')
    end
end

% PLOT MAX VALUE GRAPHS, ETC.
% check if maxvalue graph should be plotted
if plotmaxvaluepeakcheck == 1
    if mainplottype <= 3
        maxptitle = 'Max Values (Peak of curves)';
    else
        maxptitle = 'Max Values (Peak of curves) normalised
for lamp intensity variations';
    end
    maxpxaxis = 'Wavelength (nm)';
    maxpyaxis = 'Counts';

MaxValuePlotterV11(maxvaluearray,maxarray,maxptitle,maxpxaxis,
maxpyaxis)
end

% check if max values (integrated curves) should be plotted
if plotmaxvalueintegratedcheck == 1
    % define labels for graph
    if mainplottype <= 3
        maxititle = 'Max Values (Integral of the curves)';
    else
        maxititle = 'Max Values (Integral of the curves)
normalised for lamp intensity variations';
    end
    maxixaxis = 'Wavelength (nm)';
    maxiyaxis = 'Counts';
```

```
    % plot integrated normalised enhancement values as a
measure of max

MaxValuePlotterV11(intnormalen,maxarray,maxititle,maxixaxis,ma
xiyaxis)
end

% plot integrated enhancement counts
if plotintegratedlampintesitycheck == 1
    % define labels for graph
    inttitle = 'Integral of Lamp intensity at each
wavelength';
    intxaxis = 'Wavelength (nm)';
    intyaxis = 'Counts';
    % plot integrated normalised enhancement values as a
measure of max

MaxValuePlotterV11(intenhancement,maxarray,inttitle,intxaxis,i
ntyaxis)
end
```

DataExtractor.m: extracts the data and stores it into an array. (Listed previously).

DataManipulator.m: manipulates the data as required, in this case it is only used to calibrate intensity for variations in the enhancement-lamp intensity. (Listed previously).

DataSmoother.m: smoothes the data, by taking the average of a desired number of data points centred on the point it saves to. (Revised edition to work with this programme).

```
function [data] =
DataSmoother(data,datalength,smoothingsize,columntosmooth,smoo
thedcolumn);
% this function smoothes data sets to reduce noise. It
calculates the average
% of a desired number of terms (smoothingsize) and saves the
resultant
% number into a new array. It uses an odd number of terms to
smooth and
% puts the number into the middle place. beginning and end
numbers use the
```

```
% max of eachside on any side of the centre value.
%
% calling sequence: [data] =
DataSmoother(data,datalength,smoothingsize,columntosmooth)
%
% Input variables
% Data              an array of numbers, one column of which
is to be smoothed
% datalength        number of rows in the array
% smoothingsize     number of terms to be averaged
% columntosmooth    column in the array that is to be
smoothed.
% smoothedcolumn    where the smoothed data is saved
%
% Output variables
% data              data with the desired column replaced with
the smoothed one
%
% Author: Raphael Nolden © 2007

% check that an appropriate number has been entered for
smoothingsize
if smoothingsize <= 1 || rem(smoothingsize,1)~=0
    error('smoothing size must be a positive integer and
greater than 2')
end

% check smoothingsize to ensure it is an odd number otherwise
assigns a new value of specified + 1
if rem(smoothingsize,2) == 0
    smoothingsize = smoothingsize + 1;
    fprintf('the value entered for smoothingsize is not an odd
number so %2u was used instead\n',smoothingsize)
end

% extracts the desired column of data out of the data array
smoothingdata = data(:,columntosmooth);

% define how many values are on each side of the centre point
eachside = (smoothingsize - 1)/2;
backwardcounter = eachside;

% perform first 'eachside' number of iterations
for ii=1:1:eachside
    thissample = smoothingdata(1:ii+eachside,1);
    smoothdata(ii,1) = sum(thissample)/(ii+eachside);
end
```

```
% performs smoothing where smoothingsize number of variables
are available
% to do smoothing over
for jj = eachside+1:1:(datalength-eachside)
    % create subarray of smoothingsize points to average
    thissample = smoothingdata(jj-eachside:jj+eachside,1);
    % average data and save data into array
    smoothdata(jj,1) = (sum(thissample))/smoothingsize;
end

% perform smoothing on final data points
for kk=datalength-eachside+1:1:datalength
    thissample = smoothingdata(kk-eachside:datalength,1);
    smoothdata(kk,1) =
sum(thissample)/(backwardcounter+eachside);
    backwardcounter = backwardcounter - 1;
end
```

MaxFinder.m: finds the maximum value in the data set; this can be restricted to a certain domain of the data. (Listed previously).

Plotter.m: plots the data. (Listed previously).

PlotDecoratorV11.m: labels the graph and adds a legend. The names of these can be inputted into the function rather than requiring manual input from the user.

```
function [] =
PlotDecoratorV11(legendnamescell,datasetcounter,def,graphtitle
,xaxislabel,yaxislabel);
%
% plot decorator
% should be used after plotting multiple graphs. it takes
input
% from the user, for things such as title and axis labels, and
then allows
% custom axes and legends to be defined; otherwise it uses the
defaults,
% including the default legend entered as the legendnamescell
%
% calling sequence:
PlotDecoratorV11(legendnamescell,datasetcounter,def,graphtitle
,xaxislabel,yaxislabel)
%
```

```
% DEFINITION OF INPUTS
% if 2 inputs defined its the legendnamescell and
datasetcounter
% if 3 inputs defined its the legendnamescell, datasetcounter,
def
% if 4 inputs defined its the legendnamescell, datasetcounter,
def, graphtitle
% if 5 inputs defined its the legendnamescell, datasetcounter,
def, xaxislabel, yaxislabel
% if 6 inputs defined all inputs are defined
%
% if 2 inputs are defined it will ask if default legend and
axes should be
% used and ask for inputs for axes and title
% if 3 inputs are defined it will collect the axis values and
title
% if 4 inputs are defined it will collect the axis values
% if 5 inputs are defined it will collect the graph title
%
% Input variables:
% legendnamescell       cell array of legend names
% datasetcounter        number of datasets plotted
% def                   to use default axis and legend use 0
otherwise 1
% graphtitle            graph title
% xaxislabel            x axis name
% yaxislabel            y axis name
%
% NO Output variables
%
% Author: Raphael Nolden © 2007
%
% REVISIONS
% V1.1
% uses nargin and nargout to check number of inputs and only
asks for ones
% which have not been defined. also takes in values for title
and axis names
% and if default axes to be used
%



% check which arguments have not been defined and use input
statements for
% these.

% if less than 2 inputs are defined this programme cannot run
display error message
if nargin < 2
    error('ERROR not enough input arguments')
```

```
% if only 2 inputs are defined these are the legendnamescell
and datasetcounter
elseif nargin == 2
    % input and set standard graph labels
    graphtitle = input('Input title for graph: ','s');
    title(graphtitle)
    xaxislabel = input('Input x-axis label: ','s');
    xlabel(xaxislabel)
    yaxislabel = input('Input y-axis label: ','s');
    ylabel(yaxislabel)
    alldefaultcheck = input('To use default axis and legend
names press enter, otherwise press 1 then enter: ');

% if 3 inputs are not defined, these are defined as the 3 main
labels
elseif nargin == 3
    % input standard graph labels
    graphtitle = input('Input title for graph: ','s');
    xaxislabel = input('Input x-axis label: ','s');
    yaxislabel = input('Input y-axis label: ','s');

    % label graph
    title(graphtitle)
    xlabel(xaxislabel)
    ylabel(yaxislabel)
    alldefaultcheck = def;

% if two inputs are not defined, these are defined as the axis
labels
elseif nargin == 4
    % ask for inputs for axis labels
    xaxislabel = input('Input x-axis label: ','s');
    yaxislabel = input('Input y-axis label: ','s');

    % label graph
    title(graphtitle)
    xlabel(xaxislabel)
    ylabel(yaxislabel)
    alldefaultcheck = def;

% if just one input is not defined this is the title
elseif nargin == 5
    % since the title is not defined, need to shift the inputs
to correspond
    % to x and y labels
    yaxislabel = xaxislabel;
    xaxislabel = graphtitle;
```

```
    % assign values
    alldefaultcheck = def;

    % get inputs required
    graphtitle = input('Input title for graph: ','s');

    % label graph
    title(graphtitle)
    xlabel(xaxislabel)
    ylabel(yaxislabel)

% if all arguments have been input, label graph
elseif nargin == 6
    % label graph
    title(graphtitle)
    xlabel(xaxislabel)
    ylabel(yaxislabel)
    alldefaultcheck = 0;

% if too many arguments have been put in, display error
message
elseif nargin > 6
    error('ERROR to many input arguments')
end


% if default legend and axis is not selected collect relevant
data
if alldefaultcheck == 1
    % determines which parameters are to be custom
    customaxischeck = input('To enter custom axis values press
1 then enter, otherwise press enter: ');
    customlegendcheck = input('To input custom legend names
press 1 then enter, otherwise press enter: ');

    % determine type of axis desired and set
    if customaxischeck == 1
        axstr = input('Please enter the desired axis values as
a vector [xmin xmax ymin ymax]: ');
        axis([axstr])
    end

    % determine type of legend required and create it
    if customlegendcheck == 1

        % inform user how to input custom legend names
        disp('Enter one legend title at a time, in the order
they appeared in the data name file')
```

```
        % collect as many legend names as there are datasets
        for ii = 1:1:datasetcounter
            thislegend = input('Input legend name: ','s');
            legendnamescell{ii} = thislegend;
        end
    end
end

% apply legend to graph, using legend chosen or defined
earlier
legend(legendnamescell,'location','north')
legend boxoff
```

MaxValuePlotterV11.m: allows plot labels, etc., to be included in function call, enabling automated labelling.

```
function [] =
MaxValuePlotterV11(maxvaluearray,maxarray,graphtitle,xaxislabe
l,yaxislabel);
% function to plot the maxvalue array. takes in maxvalue array
and plots
% it against y-array
%
% Input:
% maxvaluearray     array of max values to be plotted
% maxyarray         array of y values
%
% Author: Raphael Nolden © 2007
%
% Revisions:
%
% Version 1.1
% takes in names of title and axes for plotting so this can be
automated

% define new figure and start hold on
figure

% plot data
plot(maxarray,maxvaluearray,'+',maxarray,maxvaluearray,'-')

% input and set standard graph labels
title(graphtitle)
xlabel(xaxislabel)
ylabel(yaxislabel)
```

APPENDIX 3: Multi-Concentration Analyser

We wrote this programme to analyse multiple emission profiles taken with different concentrations. It also allows for these to be compared, averaged, and normalised.

This programme uses some of the same functions we wrote for the other programmes. They are only shown once, at their first occurrence, to prevent repetition.

MultiConcentrationAnalyser.m: requires input of the absorption values measured using the absorption spectrometer. It also requires the name of one text file for each concentration which contains the necessary information that allows the computer to find all other required files. It extracts all the data from the .txt files and stores them into a large cell array. It then analyses this data, normalises some of it, and plots a selection of graphs.

```
% multiconcentrationanalyser
%
% this programme analyses multiple emission profiles with
different
% concentrations.
%
% Description of terms in the allexptdata cell array:
%
% the columns represent the experiments
%
% row 6 contains the concentrations
% row 5 contains the normalised peak values
% row 4 contains the peak values
% row 3 contains the integrated values
% row 2 contains the normalised integrated values
% row 1 contains the the alldata arrays
%
% each alldata array:
% the rows represent the wavelengths
```

```
% column 1 contains the pre-enhancement data
% column 2 contains the during 1 enhancement data
% column 3 contains the during 2 enhancement data
% column 4 contains the post-enhancement data
% column 5 contains the enhancement array
%
% for all the following arrays, the rows represent the
wavelength at which
% the measurement was taken
%
% the enhancement array:
% column 1 contains the wavelengths
% column 2 contains the enhancement
% column 3 contains the enhancement, normalised for changes in
lamp
% intensity
%
% the pre- and post-enhancement arrays:
% column 1 contains the wavelength (nm)
% column 2 contains the excitation (a.u.)
% column 3 contains the emission (a.u.)
% column 4 contains the std. dev. excitation
% column 5 contains the std. dev. emission
% column 6 contains the smoothed data (smoothed version of col
2)
% column 7 contains the manipulated data (col 3, normlised for
lamp intensity)
%
% the during enhancement arrays
% column 1 contains the wavelength (nm)
% column 2 contains the excitation (a.u.)
% column 3 contains the emission (a.u.)
% column 4 contains the std. dev. excitation
% column 5 contains the std. dev. emission
%
%
% OTHER ARRAYS OF DATA WHICH ARE CREATED:
%
% MAX VALUE ARRAYS
%
% % % integrated values
% allintenhancement
% % % integrated values normalised
% allnormalisedintenhancement
% % % peak values
% allpeakarray
% % % peak values normalised
% allnormalisedpeakarray
%
% % Arrays of max values normalised for concentration
```

```
% concnormalisedintmaxarray
% concnormalisednormalisedintmaxarray
% concnormalisedpeakarray
% concnormalisednormalisedpeakarray
%
% % mean of arrays normalised for concentration
% meanconcnormintmaxarray
% meanconcnormnormalisedintmaxarray
% meanconcnormpeakarray
% meanconcnormnormalisedpeakarray
%
% Author: Raphael Nolden © 2007
%

% clear memory and initialise global variables
clear all
clc
cpu1 = cputime;
global colourcounter linecounter

% input required variables
% number of experiments (i.e. number of different
concentrations
noofexpts = 4;
% number of measurements taken per wavelength
noofparts = 4;
% ensure alpha (absorption coefficient) and path length are in
the same units
alpha = 138000;
pathlength = 1;
% define wavelength vector over which these measurements were
done
lambda = [200:10:300];


% input names of text files containing file names for each
expt. (these file
% names have the names of the four file names for each
experiment. Each of
% these files should contain all of the pre, during 1, during
2, or post data)
% names for the corresponding expt.
exptlist{1} = 'EXPT1.txt';
exptlist{2} = 'EXPT3.txt';
exptlist{3} = 'EXPT4.txt';
exptlist{4} = 'EXPT5.txt';

% input absorption value measured for each expt. at the
central peak
```

```
absorption(1) = 1.31832;
absorption(2) = 0.09148;
absorption(3) = 2.3125;
absorption(4) = 0.0547976;

% calculate the concentration and save it to an array (units
Mol/L)
for iii = 1:1:noofexpts
    concentration(iii) = absorption(iii)/(alpha * pathlength);
end

% Create array of names for legend and convert units to make
them more
% useful

% convert units from Mol/L to µMol/L
legendconc = (concentration*1e9)/1e3;

% create array of legend names
for ii = 1:1:noofexpts

    % convert the re-unitted concentration to a string
    thislegendname =  num2str(legendconc(ii));

    % concatenate the concentration with the units
    legendnamescell{ii} = [thislegendname, ' (\muMol/L)'];
end

% COLLECT DATA

% loop through all expts. extract all data and return large
array
for ii = 1:1:noofexpts

    % extract expt. name
    filename = char(exptlist(ii));

    % open expt. file ready to read out separate parts
    fid = fopen(filename);

    % read in names of all files that need to be put into
function file to
    % extract data for this expt. and save into an array
    for jj = 1:1:noofparts
        thesefilenames = fgetl(fid);
        filenamearray{jj} = thesefilenames;
    end
```

```
    % run function file to do all data extraction etc
    [alldata intenhancement normalisedintenhancement peakarray
normalisedpeakarray] =
MultidataFunc(filenamearray(1),filenamearray(2),filenamearray(
3),filenamearray(4));

    % save data from expt. into an array for later use
    allexptdata{1,ii} = alldata;
    allexptdata{2,ii} = intenhancement;
    allexptdata{3,ii} = normalisedintenhancement;
    allexptdata{4,ii} = peakarray;
    allexptdata{5,ii} = normalisedpeakarray;

    % create arrays for each peak value type for simple access

    % create array of integrated values
    allintenhancement(ii,:) = intenhancement;

    % create array of normalised integrated values
    allnormalisedintenhancement(ii,:) =
normalisedintenhancement;

    % create array of peak values
    allpeakarray(ii,:) = peakarray;

    % create array of normalised values
    allnormalisedpeakarray(ii,:) = normalisedpeakarray;
end

% add concentrations to the allexptdata cellarray
for jjj = 1:1:noofexpts
    allexptdata{6,jjj} = concentration(jjj);
end


% PEAK VALUES NOT NORMALISED FOR CONCENTRATION
figure
hold on
% plot peak values
for kkk = 1:1:noofexpts
    % create array for plotter programme
    data(:,1) = lambda;
    data(:,2) = allpeakarray(kkk,:);
    Plotter(data,1,2)
end

% turn off hold command, insert a legend and reset counter
variables
```

```
hold off
legend(legendnamescell,'location','northwest')
colourcounter = 0;
linecounter = 1;
title('Peak values not normalised for concentration or
variations in enhancement ligth intensity')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')


% INTEGRATED VALUES NOT NORMALISED FOR CONCENTRATION OR
VARIATION IN
% ENHANCEMENT LAMP CONC.
figure
hold on
% plot integrated values
for kkk = 1:1:noofexpts
    % create array for plotter programme
    data(:,1) = lambda;
    data(:,2) = allintenhancement(kkk,:);
    Plotter(data,1,2)
end

% turn off hold command, insert a legend, and reset counter
variables
hold off
legend(legendnamescell,'location','northwest')
colourcounter = 0;
linecounter = 1;
title({'Emission Profile from integrated values';'Not
normalised for variations in concentration';'or enhancement
ligth intensity'})
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')


% ANALYSIS

% normalise peak values for concentration
for kk = 1:1:noofexpts
    concnormalisedpeakarray(kk,:) =
allpeakarray(kk,:)/concentration(kk);
end

figure
% plot concentration-normalised peak values
hold on
for kkk = 1:1:noofexpts
    data(:,1) = lambda;
```

```
    data(:,2) = concnormalisedpeakarray(kkk,:);
    Plotter(data,1,2)
end

% turn off hold command, insert a legend, and reset counter
variables
hold off
legend(legendnamescell,'location','north')
colourcounter = 0;
linecounter = 1;
title('Peak values normalised for concentration')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')

% normalise the lamp-intensity-normalised peak values for
concentration
for kk = 1:1:noofexpts
    concnormalisednormalisedpeakarray(kk,:) =
allnormalisedpeakarray(kk,:)/concentration(kk);
end

% plot normalised peak values
figure
hold on
for kkk = 1:1:noofexpts
    data(:,1) = lambda;
    data(:,2) = concnormalisednormalisedpeakarray(kkk,:);
    Plotter(data,1,2)
end

% turn off hold command, insert a legend, and reset counter
variables
hold off
legend(legendnamescell,'location','north')
colourcounter = 0;
linecounter = 1;
title('Peak values normalised for concentration and variation
in enhancement lamp intensity')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')

% normalise integrated max values for concentration
for kk = 1:1:noofexpts
    concnormalisedintmaxarray(kk,:) =
allintenhancement(kk,:)/concentration(kk);
end

% plot integrated max values
figure
```

```
hold on
for kkk = 1:1:noofexpts
    data(:,1) = lambda;
    data(:,2) = concnormalisedintmaxarray(kkk,:);
    Plotter(data,1,2)
end

% turn off hold command, insert a legend, and reset counter
variables
hold off
legend(legendnamescell,'location','north')
colourcounter = 0;
linecounter = 1;
title({'Emission Profile from Integrated values,';'normalised
for concentration only'})
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')

% normalise for concentration the lamp-intensity-normalised
integrated max
% values
for kk = 1:1:noofexpts
    concnormalisednormalisedintmaxarray(kk,:) =
allnormalisedintenhancement(kk,:)/concentration(kk);
end


% plot concentration-normalised, lamp intensity-normalised max
values.
figure
hold on
for kkk = 1:1:noofexpts
    data(:,1) = lambda;
    data(:,2) = concnormalisednormalisedintmaxarray(kkk,:);
    Plotter(data,1,2)
end

% turn off hold command, insert a legend, and reset counter
variables
hold off
legend(legendnamescell,'location','north')
colourcounter = 0;
linecounter = 1;
title({'Emission Profile from integrated values';'normalised
for variations in enhancement';'lamp intensity and
concentration'})
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')
```

```
% TAKING THE AVERAGE OF THE MAX VALUES

% average all the max arrays
meanconcnormpeakarray =
(sum(concnormalisedpeakarray))/noofexpts;
meanconcnormnormalisedpeakarray =
(sum(concnormalisednormalisedpeakarray))/noofexpts;
meanconcnormintmaxarray =
(sum(concnormalisedintmaxarray))/noofexpts;
meanconcnormnormalisedintmaxarray =
(sum(concnormalisednormalisedintmaxarray))/noofexpts;

% plot and label the averaged max arrays
figure
plot(lambda,meanconcnormpeakarray)
title('Mean of the peak values normalised for cocentration')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')
figure
plot(lambda,meanconcnormnormalisedpeakarray)
title('Mean of the double normalised peak values')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')
figure
plot(lambda,meanconcnormintmaxarray)
title('Mean of integrated values normalised for concentration
only')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')
figure
plot(lambda,meanconcnormnormalisedintmaxarray)
title({'Mean of Integrated values normalised for';'variations
in enhancement lamp intensity and concentration'})
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')

% MEAN VALUES COMPENSATED FOR LAMP-INTENSITY VARIATIONS AFTER
AVERAGING

% extract and sum enhancement light for all wavelengths in all
expts.
for jj = 1:1:length(lambda)
    for ii = 1:1:noofexpts
        sumenhanvmentlight(ii,jj) =
sum(allexptdata{1,ii}{jj,2}(:,2)) +
sum(allexptdata{1,ii}{jj,3}(:,2));
    end
end
```

```
% calculate the total and mean enhancement light for all
wavelengths
totalenhanvementlight = sum(sumenhanvmentlight);
meantotalenhanvementlight = totalenhanvementlight/noofexpts;

% normalise for enhancement-lamp intensities after summation
and
% normalisation for concentration
latenormalisedpeaks =
meanconcnormpeakarray./meantotalenhanvementlight;
latenormalisedint =
meanconcnormintmaxarray./meantotalenhanvementlight;

% plot and label graphs from the above arrays
figure
plot(lambda,latenormalisedpeaks)
title('Peaks normalised for enhancement lamp intensity
variation after averaging etc')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')
figure
plot(lambda,latenormalisedint)
title('Integrated values normalised for enhancement lamp
intensity variation after averaging etc')
xlabel('Wavelength (nm)')
ylabel('Intensity (a.u.)')

% PLOTTING INTENSITY vs. CONCENTRATION FOR EVERY WAVELENGTH
% re-order the concentration array for plotting
newconc(1) = concentration(4);
newconc(2) = concentration(2);
newconc(3) = concentration(1);
newconc(4) = concentration(3);

% re-order peak value array to match with above
newmax(1,:) = allpeakarray(4,:);
newmax(2,:) = allpeakarray(2,:);
newmax(3,:) = allpeakarray(1,:);
newmax(4,:) = allpeakarray(3,:);


% plot concentration against intensity for all wavelengths
figure
hold on
plarray(:,1) = newconc;
for ii = 1:1:length(lambda)
    plarray(:,2) = newmax(:,ii);
    Plotter(plarray,1,2)
end
```

```
% turn off hold command and reset counter variables
hold off
colourcounter = 1;
linecounter = 1;

% label graph
title('The effects of enhancement wavelength and concentration
intensity')
xlabel('Concentration (Mol/L)')
ylabel('Intensity (a.u.)')

% create a cell array of legend names
for ii = 1:1:length(lambda)
    lambdalegendnamescell{ii} = num2str(lambda(ii));
end

% add legend to the plot
legend(lambdalegendnamescell,'location','northwest')

% semilog plot concentration against intensity for all
wavelengths
figure
semilogx(newconc,newmax(:,1))
hold on
for ii = 2:1:length(lambda)
    plarray(:,2) = newmax(:,ii);
    Plotter(plarray,1,2)
end

% turn off hold command and reset counter variables
hold off
colourcounter = 1;
linecounter = 1;

% label graph and add legend
title('The effects of enhancement wavelength and concentration
intensity')
xlabel('Concentration (Mol/L)')
ylabel('Intensity (a.u.)')
legend(lambdalegendnamescell,'location','northwest')

% calculate the mean value for each concentration
Tnewmax = newmax';
meanen = (sum(Tnewmax))/length(lambda);

% convert units on sorted conc to µMol/L
newunitnewconc = (newconc*1e9)/1e3;
```

```
% make array of these mean values
meandata(1,:) = newunitnewconc;
meandata(2,:) = meanen;

% define the uncertainties
uncert = 2e4;

% add uncertainty values to this array
for ii = 1:1:noofexpts
    meandata(3,ii) = uncert;
end

% plot graph of mean values
SinglePlotter(meandata,1,2,3);

% label the graph
title('Mean Intensity from each wavelength at a given
concentration')
xlabel('Concentration (\muMol/L)')
ylabel('Intensity (a.u.)')

% semilog plot mean values graphs
figure
semilogx(newconc,meanen)
title('Mean of all expt data for each concentration')
xlabel('Concentration (Mol/L)')
ylabel('Intensity (a.u.)')

% normalise this mean intensity for concentration
normalisedmeanen = meanen./newconc;

% make array of these mean values
meandata(1,:) = newconc;
meandata(2,:) = normalisedmeanen;

% define the uncertainties
uncert = 5e9;

% add uncertainty values to this array
for ii = 1:1:noofexpts
    meandata(3,ii) = uncert;
end

% plot the concentration-normalised version of above graph
SinglePlotter(meandata,1,2,3);

% label the graph
```

```
title('Mean wavelength data corrected for concentration')
xlabel('Concentration (Mol/L)')
ylabel('Intensity (a.u.)')

% semilog plot the concentration-normalised version of above
graph
figure
semilogx(newconc,normalisedmeanen)
title('Mean of all expt data for each concentration normalised
for concentration')
xlabel('Concentration (Mol/L)')
ylabel('Intensity (a.u.)')

% calculate CPU time used to run this data
cpu2 = cputime;
runtime = cpu2-cpu1
```

MultiDataFunc.m: is based on the single-concentration analysis programme; however, it has had all sections for plotting, etc., removed. We also removed the user data input and other unnecessary parts.

```
function [alldata intenhancement normalisedintenhancement
peakarray normalisedpeakarray] =
MultidataFunc(prenamefile,during1namefile,during2namefile,post
namefile);
% Author: Raphael Nolden © 2007
%
% based on Version 2.2.2 of ExcitationProfileAnalyserV222.m
from the single-concentration
% emission profile analysis programme
%
% uses excitation values to compensate for variations in lamp
% intensity
%
% Input variables
% prenamefile         cell containing the name of the pre-
excitation file
% during1namefile     cell containing the name of the first
excitation file
% during2namefile     cell containing the name of the second
excitation file
% postnamefile        cell containing the name of the post-
excitation file
%
% Output Data Arrays
% alldata
% intenhancement
```

```
% normalisedintenhancement
% peakarray
% normalisedpeakarray


% INITIALISE VARIABLES

% initialise global variables
global colourcounter linecounter

% initialise counters
colourcounter = 0;
linecounter = 1;
datasetcounter = 0;

% INITIALISE VARIABLES BY HARD-CODING THEM IN
datasmoothercheck = 1;
noofcolumns = 5;
headerlength = 23;
wavelengthcol = 1;
enhancementcol = 3;
usefulrange = 0;
smoothingsize = 9;
columntosmooth = 2;
smoothedcolumn = 6;
plotmaxvalueintegratedcheck = 1;
plotmaxvaluepeakcheck = 1;
plotintegratedlampintesitycheck = 1;
preandpostcheck = 0;
manipsavetocol = 7;
emissioncol = 3;

% convert the name files to characters from cells
prenamefile = char(prenamefile);
during1namefile = char(during1namefile);
during2namefile = char(during2namefile);
postnamefile = char(postnamefile);

% define files which contain the names of the data files for
each type
fidpre = fopen(prenamefile);
fidduring1 = fopen(during1namefile);
fidduring2 = fopen(during2namefile);
fidpost = fopen(postnamefile);

% PERFORM DATA EXTRACTION MANIPULATION
```

```
% Extract filenames from the file until all have been read and
their data
% has been extracted
while 1
    % select the next filename of each type to extract data
from
    thisprename = fgetl(fidpre);
    thisduring1name = fgetl(fidduring1);
    thisduring2name = fgetl(fidduring2);
    thispostname = fgetl(fidpost);

    % check if end of file containing names of data files has
been reached
    if thisprename == -1
        break
    elseif thisduring1name == -1
        break
    elseif thisduring2name == -1
        break
    elseif thispostname == -1
        break
    end

    % count number of data sets plotted to enable custom
legend to be made
    datasetcounter = datasetcounter + 1;

    % run data extraction function for each of the four data
files and save
    % the data into four different arrays
    [predata,predatalength] =
DataExtractor(thisprename,headerlength,noofcolumns);
    [during1data,duringdatalength] =
DataExtractor(thisduring1name,headerlength,noofcolumns);
    [during2data,duringdatalength] =
DataExtractor(thisduring2name,headerlength,noofcolumns);
    [postdata,postdatalength] =
DataExtractor(thispostname,headerlength,noofcolumns);

    % run the datamanipulator programme which divides the
emission data by
    % smoothed excitation data
    [predata] = DataManipulator(predata,manipsavetocol);
    [postdata] = DataManipulator(postdata,manipsavetocol);

    % check if datasmoother should be run
    if datasmoothercheck == 1
        % run the datasmoother on the pre- and post-file,
storing the smoothed data
```

```
        % into the smoothed data column
        [predata] =
DataSmoother(predata,predatalength,smoothingsize,columntosmoot
h,smoothedcolumn);
        [postdata] =
DataSmoother(postdata,postdatalength,smoothingsize,columntosmo
oth,smoothedcolumn);
    end

    % create a new array to save the enhancement values into
    % save in wavelength
    enhancement(:,1) = predata(:,1);

    % calculate the enhancement and save into enhancement
array
    % (enhancement = (post – pre), both adjusted for intensity
of lamp)
    enhancement(:,2) = postdata(:,emissioncol) -
predata(:,emissioncol);

    % integrate the enhancement light and saves to an array
    intenhancement(datasetcounter) = (sum(during1data(:,2)) +
sum(during2data(:,2)));

    % calculate the normalised enhancement
    enhancement(:,3) =
enhancement(:,2)./intenhancement(datasetcounter);

    % integrate enhancement (max integration of curve)
    intenhancement(datasetcounter) = sum(enhancement(:,2));

    % integrate normalised enhancement
    normalisedintenhancement(datasetcounter) =
sum(enhancement(:,3));

    % find peak value

    % Run the max value finder (peak value of curve)
    [maxvalue] =
MaxFinder(enhancement,2,usefulrange,predatalength);

    % save maxvalue into array
    peakarray(datasetcounter) = maxvalue;

    % find normalised peak value

    % Run the max value finder (peak value of curve)
```

```
    [maxvalue] =
MaxFinder(enhancement,3,usefulrange,predatalength);

    % save maxvalue into array
    normalisedpeakarray(datasetcounter) = maxvalue;

    % save all data in to a cell array
    alldata{datasetcounter,1} = predata;
    alldata{datasetcounter,2} = during1data;
    alldata{datasetcounter,3} = during2data;
    alldata{datasetcounter,4} = postdata;
    alldata{datasetcounter,5} = enhancement;
end
```

SinglePlotter.m: plots single data sets and has the ability to plot the associated uncertainties as well. It also finds the line of best fit and plots that onto the same graph.

```
function [] = SinglePlotter(data,xrow,yrow,uncertrow);
% plotter that plots a single dataset, adds error bars and
puts in a
% line of best fit.
%
% takes in a an array of values containing the data and
uncertainties and
% calculates the line of best fit and plots this as well.
%
% calling sequence: SinglePlotter(data,xrow,yrow,uncertrow)
%
% Input variables:
% data          data array
% xrow          row number to be used as the x-axis
% yrow          row number to be used as the y-axis
% uncertrow     row number containing the uncertainties enter
zero if no
% uncertainties are to be plotted
%
% NO Output variable
%
% Author: Raphael Nolden © 2007


% initialise figure and turn hold on
figure
hold on
```

```
% plot the data and uncertainties

% check if error bars are to be plotted
if uncertrow > 0
    % plot error bar graph
    errorbar(data(xrow,:),data(yrow,:),data(uncertrow,:),'+')
else
    % plot normal plot
    plot(data(xrow,:),data(yrow,:),'O')
end

% add line of best fit

% find the polynomial coefficients for the fit
pcoeff=polyfit(data(xrow,:),data(yrow,:),1);

% create array of numbers to plot against
xfit =
min(data(xrow,:)):range(data(xrow,:))/500:max(data(xrow,:));

% calculate the line of best fit
yfit=polyval(pcoeff,xfit);

% plot the line of best fit
plot(xfit,yfit,'m')

% turn off hold
hold off
```

The following functions are also used, but not listed again as they are the same as those of the previous programme.

DataExtractor.m, DataManipulator.m, DataSmoother.m, MaxFinder.m, Plotter.m, PlotDecoratorV11.m, and MaxValuePlotterV11.m.